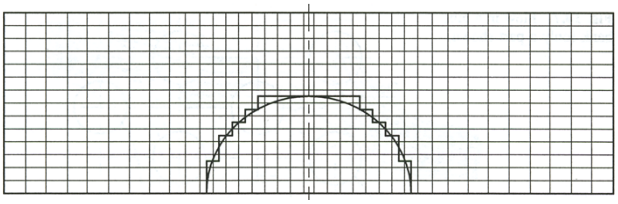
Chapter 11:
Unstructured Grids

Ibrahim Sezai
Department of Mechanical Engineering
Eastern Mediterranean University

Fall 2015-2016

Introduction

- When the solution domain is not rectangular Cartesian grids cannot be used.
- When the boundary does not coincide with the co-ordinate lines we can use stepwise approximation near the boundary.



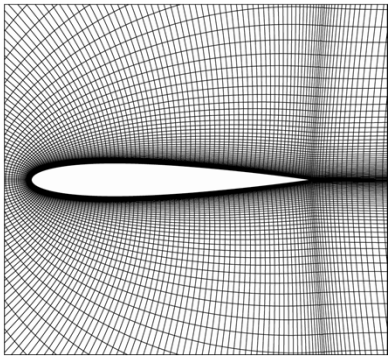
- Such an approximation is tedious and time consuming.
- Introduces errors in the computation of wall shear stresses and heat transfer.
- Cells in the solid cylinder do not take part in the calculations → waste of computer storage.

Body fitted grids

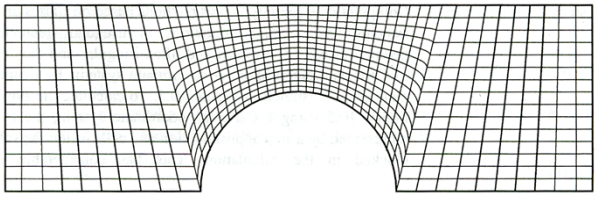
- CFD methods for complex geometries:
  1. Structured curvilinear grids (body fitted grids)
  2. Unstructured grids
- Structured curvilinear grids are based on mapping of the flow domain onto a computational domain.
- It is very difficult to find viable mappings when the geometry is complex.
- In such cases the domain may be sub-divided into several sub-regions or blocks. → Block structured grids.
- For the most complex geometries unstructured grids are used.
- Most commercial packages use unstructured grids.

Body fitted grids

An example of an orthogonal curvilinear mesh for calculating flow around an aerofoil

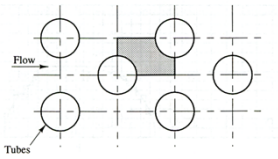


Use of a non-orthogonal body-fitted grid arrangement for the prediction of flow over a half-cylinder

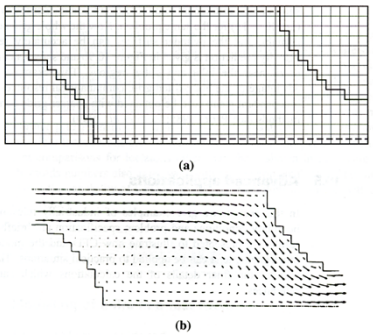


Cartesian vs. curvilinear grids

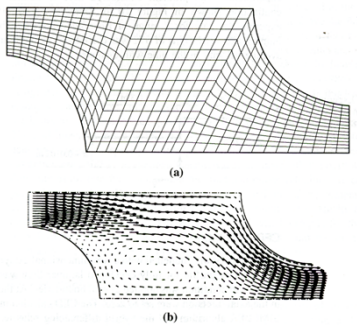
Flow over a heat exchanger tube bank (only a part shown) →



Body fitted grids give better results with no waste of grids



(a) Cartesian grid using an approximated profile to represent cylindrical surfaces;
(b) predicted flow pattern using a 40 X 15 Cartesian grid



(a) Non-orthogonal body-fitted grid for the same problem;
(b) predicted flow pattern using a 40 X 15 structured body-fitted grid

Curvilinear grids - Difficulties

- The governing equations in curvilinear co-ordinate system are much more complex. The equations in 2-D are:

Continuity

$$\frac{\partial}{\partial\xi}(\rho U)+\frac{\partial}{\partial\eta}(\rho V)=0 \qquad (11.1)$$

x-momentum

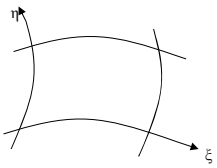
$$J\frac{\partial(\rho u)}{\partial t}+\frac{\partial}{\partial\xi}\left[\rho Uu-\mu Jg^{11}u_\xi\right]+\frac{\partial}{\partial\eta}\left[\rho Vu-\mu Jg^{22}u_\eta\right] \qquad (11.2)$$
$$=-y_\eta\frac{\partial p}{\partial\xi}+y_\xi\frac{\partial p}{\partial\eta}+\frac{\partial}{\partial\xi}\left[-\mu Jg^{12}u_\eta\right]+\frac{\partial}{\partial\eta}\left[-\mu Jg^{12}u_\xi\right]+JS_u$$

y-momentum

$$J\frac{\partial(\rho\upsilon)}{\partial t}+\frac{\partial}{\partial\xi}\left[\rho U\upsilon-\mu Jg^{11}\upsilon_\xi\right]+\frac{\partial}{\partial\eta}\left[\rho V\upsilon-\mu Jg^{22}\upsilon_\eta\right] \qquad (11.3)$$
$$=+x_\eta\frac{\partial p}{\partial\xi}-x_\xi\frac{\partial p}{\partial\eta}+\frac{\partial}{\partial\xi}\left[-\mu Jg^{12}\upsilon_\eta\right]+\frac{\partial}{\partial\eta}\left[-\mu Jg^{12}\upsilon_\xi\right]+JS_\upsilon$$

Metric coefficients

$$\begin{cases} g^{12}=-(x_\xi x_\eta+y_\xi y_\eta)/J^2 \\ g^{11}=(x_\eta^2+y_\eta^2)/J^2 \\ g^{22}=(x_\xi^2+y_\xi^2)/J^2 \end{cases}$$



Jacobian of transformation

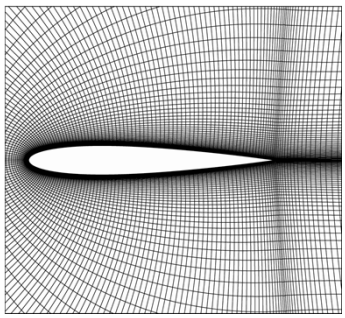
$$J=x_\xi y_\eta-x_\eta y_\xi$$

Contravariant velocities

$$U=y_\eta u-x_\eta\upsilon \qquad V=x_\xi\upsilon-y_\xi u$$

Curvilinear grids - difficulties

- Body-fitted grids are structured, so grid refinement is generally not local.



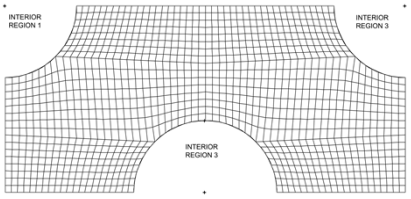
- The refinement needed to resolve the boundary layers persists in the interior mesh, which represents a waste of computer storage.

Curvilinear grids - difficulties

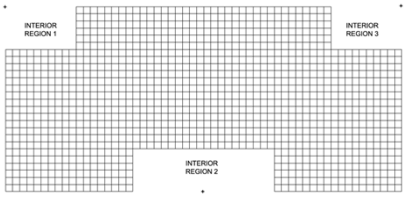
- To generate curvilinear meshes it is necessary to map the physical geometry into a computational geometry.

Mapping of physical geometry to computational geometry in structured meshes.→

Mapping may be a tedious task if the domain is more complex.



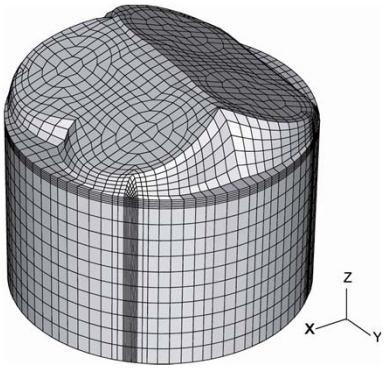
(a) physical grid in *x-y* co-ordinates



(b) the mapped structure for (a) in the computational domain

Curvilinear grids - difficulties

- For complex geometries structured grid generation may be very difficult resulting in skewed cells, which can lead to stability problems for CFD solvers.



A structured non-orthogonal mesh for a pent-roof i.c. engine geometry

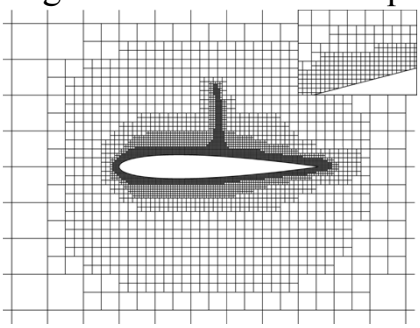
Curvilinear grids - difficulties

Problems encountered with structured grids:

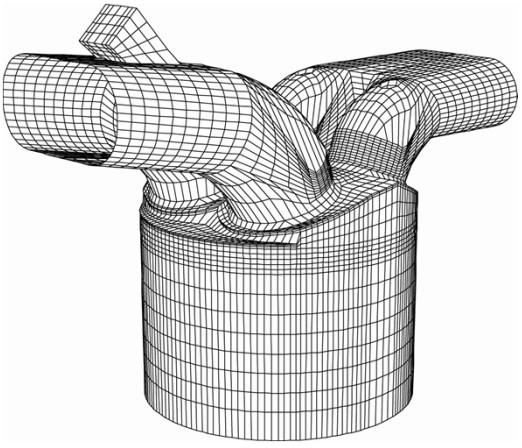
- Still difficult and time consuming to generate
- If the geometry cannot be readily mapped into a rectangle (in 2D) or parallellepiped (3D) this can result in skewed grid lines.
- Unnecessary grid resolutions can result when mapping is difficult
- Mapping is sometimes impossible with complex 3D geometries with internal objects/parts.

Block-structured grids

- In block-structured grids the domain is sub-divided into regions, each of which has a structured mesh.
- Such meshes are more flexible than (single block) structured meshes.
- The block structured approach allows the use of fine grids in regions where greater resolution is required.



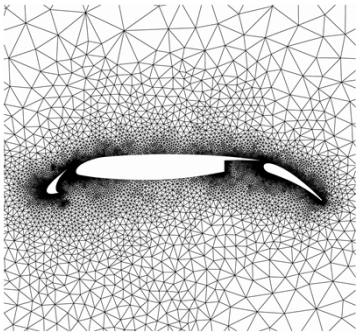
Block-structured grids



Block-structured mesh arrangement for an engine geometry, including inlet and exhaust ports, used in engine simulations with KIVA-3V

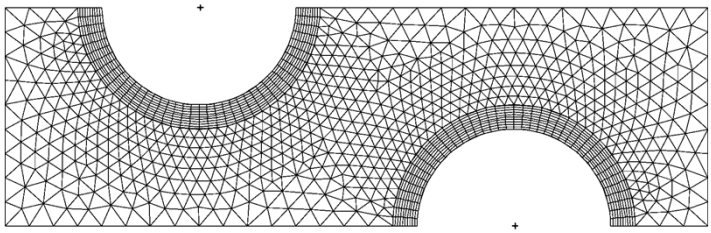
Unstructured grids

- There is no implicit structure of co-ordinate lines in unstructured grids.
- The mesh can easily be concentrated where necessary without wasting computer storage.
- Control volumes can be of any shape.



Unstructured grids

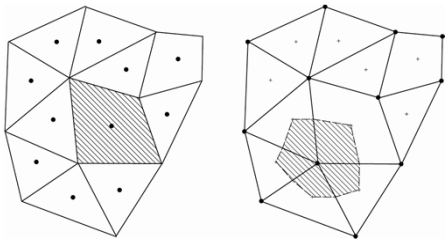
- A mixture of different cell shapes can be used.



- Unstructured mesh generation is fairly straightforward.
- Complex domains can be meshed.
- Mesh refinement and adaption (in regions with high gradients) are much more easier.

Discretisation in unstructured grids

- There are two ways of defining CV's in unstructured grids:
  1. Cell-centered control volumes
  2. Vertex-centered control volumes.



(a) cell-centred control volumes (b) vertex-centred control volumes

- We will use cell-centered method.

Discretisation in unstructured grids

The general transport equation (2.39) is

$$\frac{\partial(\rho\phi)}{\partial t} + div(\rho\phi\mathbf{v}) = div(\Gamma\, grad\,\phi) + s_\phi \qquad (11.4)$$

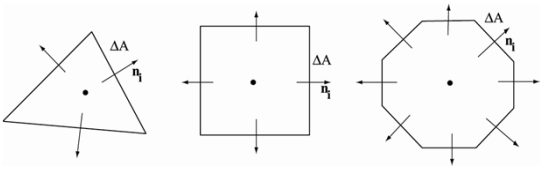
Integrating Eq. (11.4) over a 3-D control volume

$$\int_{CV}\frac{\partial(\rho\phi)}{\partial t}dV + \int_{CV} div(\rho\phi\mathbf{v})dV = \int_{CV} div(\Gamma\, grad\,\phi)dV + \int_{CV} s_\phi dV \qquad (11.5)$$

For a vector **a** Gauss' divergence theorem states

$$\int_{CV} div\,\mathbf{a}dV = \int_{A}\mathbf{n}\cdot\mathbf{a}dA \qquad (11.6)$$

n.a → component of vector **a** in the direction of the outward unit vector **n** normal to infinitesimal surface element dA.



Discretisation in unstructured grids

Application of Gauss' divergence theorem to equation (11.4) gives

$$\frac{\partial}{\partial t}\left(\int_{CV}\rho\phi dV\right)+\int_{A}\mathbf{n}\cdot(\rho\phi\mathbf{v})dA=\int_{A}\mathbf{n}\cdot(\Gamma\, grad\,\phi)dA+\int_{CV}s_{\phi}dV \qquad (11.7)$$

Using $\mathbf{A}=A\mathbf{n}$, where $\mathbf{A}=A_x\mathrm{i}+A_y\mathrm{j}+A_z\mathrm{k}$, $A=|\mathbf{A}|=(A_x^2+A_y^2+A_z^2)^{1/2}$

$\mathbf{n}=n_x\mathrm{i}+n_y\mathrm{j}+n_z\mathrm{k}$ (unit vector normal to surface)

$\mathbf{A}$ = surface area vector

A = magnitude of the surface area

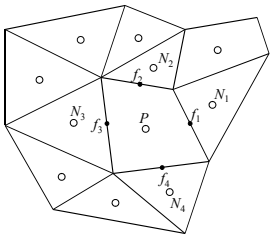
$$\underbrace{\frac{\partial}{\partial t}\left(\int_{CV}\rho\phi dV\right)}_{\text{Transient term}}+\underbrace{\oint_{A}(\rho\phi\mathbf{v})\cdot d\mathbf{A}}_{\text{Convection term}}=\underbrace{\oint_{A}(\Gamma\, grad\,\phi)\cdot d\mathbf{A}}_{\text{Diffusion term}}+\underbrace{\int_{CV}s_{\phi}dV}_{\text{Source term}} \qquad (11.8)$$

Let us discretize each term separately

Transient term

Using first order time derivative,

$$\frac{\partial}{\partial t}\left(\int_{CV}\rho\phi dV\right)\approx\frac{(\rho\phi V)_P-(\rho\phi V)_P^o}{\Delta t}$$



Superscript o refers to previous time step

Terms with no superscript refer to present time.

Cell volume V_P can be calculated by applying Gauss theorem on the identity $1=\mathrm{div}(x\mathbf{i})$:

$$V=\int_V dV=\int_V div(x\mathbf{i})dV=\int_A x\mathbf{i}\cdot\mathbf{n}dA\approx\sum_f x_f A_f^x \qquad (11.9)$$

$f\rightarrow$ refers to cell faces. For the cell of node P the summation is over the faces f_1, f_2, f_3 and f_4.

A_f^x = x-component of the cell face surface vector.

x_f = x-component of the position vector of the center of face f

Volumes and surface areas

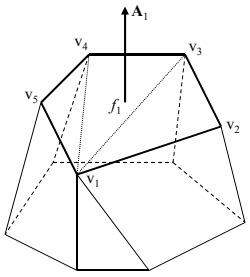
$$\mathbf{A}_f = A_f\mathbf{n} = A_f^x\mathbf{i} + A_f^y\mathbf{j} + A_f^z\mathbf{k} \tag{11.10}$$

Instead of $x\mathbf{i}$, one can also use $y\mathbf{j}$ or $z\mathbf{k}$, that is:

$$V \approx \sum_f y_f A_f^y \approx \sum_f z_f A_f^z \tag{11.11}$$

Surface areas

Consider the cell of node P. Let the top surface be f_1 with vertices v_1, v_2, v_3, v_4, v_5.



Calculation of area of surface f_1:

Decompose the surface f_1 into triangles with one common vertex. For example choose common vertex to be v_1.

The surface normal vector of the whole cell face f_1 is the sum of the surface vectors of all triangles.

$$\mathbf{A}_1 = \frac{1}{2}\sum_{i=3}^{N_v}\left[(\mathbf{r}_{i-1} - \mathbf{r}_1)\times(\mathbf{r}_i - \mathbf{r}_1)\right] \tag{11.12}$$

N_v = number of vertices in the cell face (5 vertices for face f_1)

$\mathbf{r}_i$ = position vector of vertex i

Note that there are $(N_v - 2)$ triangles. Also note that the numbering of the vertices is counter-clockwise when looking from outside the cell.

- The cell face center can be found by averaging the coordinates of the vertices of the face.

$$(x_c)_f = \frac{1}{N_v}\sum_{i=1}^{N_v} x_i \qquad (11.13)$$

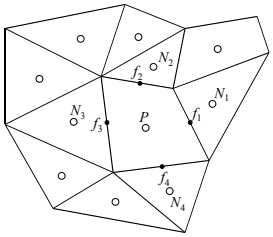
$$(y_c)_f = \frac{1}{N_v}\sum_{i=1}^{N_v} y_i \qquad (11.14)$$

$$(z_c)_f = \frac{1}{N_v}\sum_{i=1}^{N_v} z_i \qquad (11.15)$$

Diffusion term

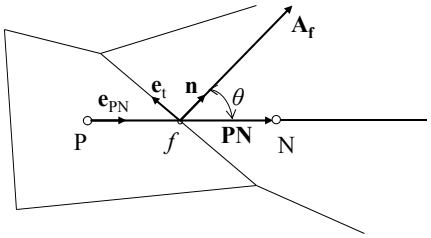
The diffusion term is approximated as

$$\oint_A (\Gamma\, grad\,\phi)\cdot d\mathbf{A} \approx \sum_{f=nb(P)} \Gamma_f \nabla\phi_f . \mathbf{A}_f \qquad (11.17)$$



where the summation is over the faces of a cell. (over the faces f_1, f_2, f_4, f_4 for cell P in the figure).

Define: $\mathbf{PN} = \mathbf{r_N} - \mathbf{r_P}$ (11.18)



$\mathbf{r_P}$, $\mathbf{r_N}$: position vector of centroids of cells P and N

$\mathbf{e}_{PN}$: unit vector along line PN

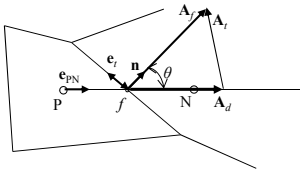
$\mathbf{n}$: unit normal vector of surface f

$\mathbf{A}_f$: surface area vector

We wish to approximate the diffusive flux $D_f = \Gamma_f \nabla\phi_f \cdot \mathbf{A}_f$ at the surface

Diffusion term

The gradient expression should involve centroids P and N of the cells.



By definition $\mathbf{e}_{PN} = \dfrac{\mathbf{PN}}{d_{PN}} \qquad d_{PN} = |\mathbf{PN}|$ (11.19)

The gradient in the direction of **e** can be written as

$$\nabla\phi_f . \mathbf{e}_{PN} = \frac{\partial\phi_f}{\partial e_{PN}} = \frac{\phi_N - \phi_P}{d_{PN}} \tag{11.20}$$

If the surface vector $\mathbf{A}_f$ is written as the sum of two vectors $\mathbf{A}_d$ and $\mathbf{A}_t$

$$\mathbf{A}_f = \mathbf{A}_d + \mathbf{A}_t \tag{11.21}$$

where $\mathbf{A}_d$ is in the direction of line PN, then

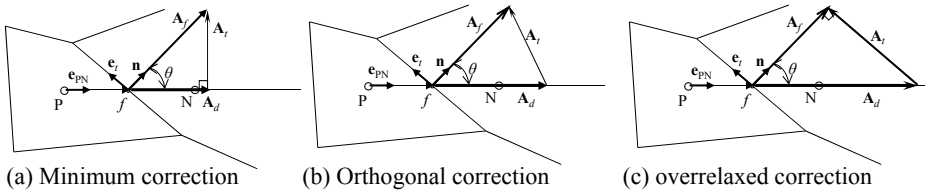
$$\nabla\phi_f . \mathbf{A}_f = \nabla\phi_f . \mathbf{A}_d + \nabla\phi_f . \mathbf{A}_t = |\mathbf{A}_d| \nabla\phi_f . \mathbf{e}_{PN} + \nabla\phi_f . \mathbf{A}_t \tag{11.22}$$

$$\nabla\phi_f . \mathbf{A}_f = \underbrace{|\mathbf{A}_d| \frac{\phi_N - \phi_P}{d_{PN}}}_{\text{orthogonal term}} + \underbrace{\nabla\phi_f . \mathbf{A}_t}_{\substack{\text{non-orthogonal} \\ \text{(cross diffusion) term}}} \tag{11.23}$$

Cross diffusion term is treated as source term.

Diffusion term

Several options can be used in defining the direction of $\mathbf{A}_t$.



(a) Minimum correction (b) Orthogonal correction (c) overrelaxed correction

(a) Set $\mathbf{A}_t$ to be perpendicular to $\mathbf{A}_d$ (minimum correction).

$$\mathbf{A}_d = A_f \cos\theta \mathbf{e}_{PN} = (\mathbf{A}_f . \mathbf{e}_{PN}) \mathbf{e}_{PN} \tag{11.24}$$

$$\mathbf{A}_t = \mathbf{A}_f - \mathbf{A}_d = A_f (\mathbf{n} - \cos\theta \mathbf{e}_{PN}) = \mathbf{A}_f - (\mathbf{A}_f . \mathbf{e}_{PN}) \mathbf{e}_{PN} \tag{11.25}$$

(b) Set $\mathbf{A}_t$ such that $|\mathbf{A}_d|$ is equal to $|\mathbf{A}_f|$ (orthogonal correction).

$$\mathbf{A}_d = A_f \mathbf{e}_{PN} \qquad \mathbf{A}_t = \mathbf{A}_f - \mathbf{A}_d = A_f (\mathbf{n} - \mathbf{e}_{PN}) \tag{11.26}$$

(c) Set $\mathbf{A}_t$ to be parallel to $\mathbf{e}_t$ (overrelaxed correction).

$$\mathbf{A}_d = \frac{A_f}{\cos\theta} \mathbf{e}_{PN} = \frac{\mathbf{A}_f \cdot \mathbf{A}_f}{\mathbf{A}_f \cdot \mathbf{e}_{PN}} \mathbf{e}_{PN} \tag{11.27}$$

$$\mathbf{A}_t = \mathbf{A}_f - \mathbf{A}_d = \mathbf{A}_f - \frac{A_f}{\cos\theta} \mathbf{e}_{PN} = \mathbf{A}_f - \frac{\mathbf{A}_f \cdot \mathbf{A}_f}{\mathbf{A}_f \cdot \mathbf{e}_{PN}} \mathbf{e}_{PN} \tag{11.28}$$

Diffusion term

Substituting the terms in Eq.(11.28) for the overrelaxed approach into Eq. (11. 23)

$$\nabla\phi_f.\mathbf{A}_f = \underbrace{A_d\frac{\phi_N-\phi_P}{d_{PN}}}_{\text{orthogonal term}} + \underbrace{\overline{(\nabla\phi)}_f.\mathbf{A}_t}_{\substack{\text{non-orthogonal}\\ \text{(cross diffusion) term}}} \qquad (11.29)$$

where $A_d = |\mathbf{A}_d| = \dfrac{\mathbf{A}_f\cdot\mathbf{A}_f}{\mathbf{A}_f\cdot\mathbf{e}_{PN}}$ $\qquad \mathbf{A}_t = \mathbf{A}_f - A_d\mathbf{e}_{PN}$

The overbar term in Eq. (11.29) is obtained via linear interpolation from the adjacent nodal values:

$$\overline{\nabla\phi}_f = (1-f_P)\nabla\phi_P + f_P\nabla\phi_N \qquad (11.30)$$

$f_P = \dfrac{\overline{Pf}}{\overline{PN}}$ or preferably: $f_P = \dfrac{\mathbf{Pf}\cdot\mathbf{e}_{PN}}{\overline{PN}} = \dfrac{\mathbf{Pf}\cdot\mathbf{e}_{PN}}{(\mathbf{PN}\cdot\mathbf{PN})^{1/2}}$ Linear interpolation factor

A more accurate approximation is to use Eq. (11.33)

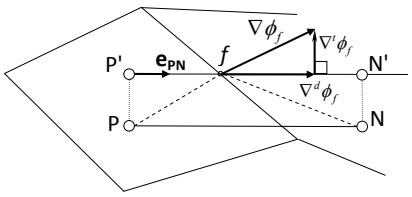
Note that A_d is the magnitude of area vector $\mathbf{A}_d$. Also note that $\mathbf{A}_d$ is in the direction of PN and $\mathbf{A}_t$ is perpendicular to $\mathbf{A}_f$.

Gradient at a Cell Face

Gradient vector of a variable ϕ at a cell face can be decomposed into two components along directions d and t, where d is along PN and t is tangential (perpendicular) to d.

$$\nabla\phi_f = \nabla^d\phi_f + \nabla^t\phi_f \qquad (11.31)$$

$$\nabla^d\phi_f = \frac{\phi_{N'}-\phi_{P'}}{|\mathbf{r}_{PN}|}\mathbf{e}_{PN} = \frac{\phi_N-\phi_P}{|\mathbf{r}_{PN}|}\mathbf{e}_{PN} + \frac{(\nabla\phi_N-\nabla\phi_P)\cdot\mathbf{PP'}}{|\mathbf{r}_{PN}|}\mathbf{e}_{PN} \qquad (11.32a)$$



where $\phi_{P'} = \phi_P + \nabla\phi_P\cdot\mathbf{PP'}$, $\phi_{N'} = \phi_N + \nabla\phi_N\cdot\mathbf{NN'}$, $\mathbf{NN'} = \mathbf{PP'}$

$$\nabla^d\phi_f = (\nabla\phi_f\cdot\mathbf{e}_{PN})\mathbf{e}_{PN}$$

$$\nabla^t\phi_f = \nabla\phi_f - \nabla^d\phi_f = \nabla\phi_f - (\nabla\phi_f\cdot\mathbf{e}_{PN})\mathbf{e}_{PN} \qquad (11.32b)$$

Substituting Eq. (11.32a) and (11.32b) into Eq. (11.31),

$$\nabla\phi_f = \frac{\phi_N-\phi_P}{|\mathbf{r}_{PN}|}\mathbf{e}_{PN} + \overline{\nabla\phi_f} - (\overline{\nabla\phi_f}\cdot\mathbf{e}_{PN})\mathbf{e}_{PN} + \frac{(\nabla\phi_N-\nabla\phi_P)\cdot\mathbf{PP'}}{|\mathbf{r}_{PN}|}\mathbf{e}_{PN} \qquad (11.33)$$

$\overline{\nabla\phi_f}$ is obtained via linear interpolation from adjacent nodal values using Eq. (11.30)

Diffusion term

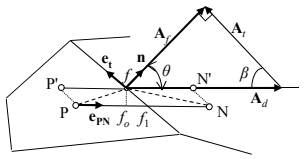
Orthogonal term is treated implicitly, whereas the cross diffusion term is treated as a source term in a deferred correction approach.

Overrelaxed approach (method (c)) is preferred, because the coefficient of the implicit term is greater than that of methods (a) and (b), and it increases with the skew angle θ. Thus, the diagonal dominance of the coefficient matrix is enhanced.

Correction for the Diffusion Term

If line PN does not pass through the face center, then a correction is needed for diffusion flux for higher accuracy.

$$\begin{aligned}\nabla\phi_f.\mathbf{A}_f &= \nabla\phi_f.\mathbf{A}_d + \nabla\phi_f.\mathbf{A}_t \\ &= |\mathbf{A}_d|\nabla\phi_f.\mathbf{e}_{PN} + \nabla\phi_f.\mathbf{A}_t \\ &= A_d\frac{\phi_{N'}-\phi_{P'}}{d_{PN}} + \nabla\phi_f.\mathbf{A}_t\end{aligned}$$



using $\phi_{P'} = \phi_P + \nabla\phi_P \cdot \mathbf{PP}'$, $\phi_{N'} = \phi_N + \nabla\phi_N \cdot \mathbf{NN}'$, $\mathbf{NN}' = \mathbf{PP}'$

$$\nabla\phi_f.\mathbf{A}_f = \underbrace{\frac{\phi_N-\phi_P}{d_{PN}}A_d}_{\text{orthogonal term}} + \underbrace{\overline{\nabla\phi_f}.\mathbf{A}_t + \frac{(\nabla\phi_N-\nabla\phi_P)\cdot\mathbf{PP}'}{d_{PN}}A_d}_{\text{non-orthogonal (cross diffusion term)}} \qquad (11.34)$$

where $\mathbf{PP}' = |\mathbf{f_1f}|\mathbf{e_t}$, $\sin\beta = \frac{|\mathbf{f_of}|}{|\mathbf{f_1f}|} = \frac{A_f}{A_d} \rightarrow |\mathbf{f_1f}| = |\mathbf{f_of}|\frac{A_d}{A_f}$,

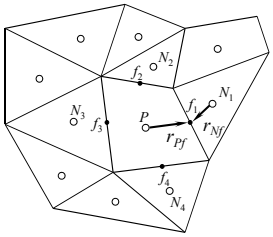
$$\mathbf{f}_o\mathbf{f} = \mathbf{Pf} - \mathbf{Pf}_o = \mathbf{Pf} - (\mathbf{Pf}\cdot\mathbf{e_{PN}})\mathbf{e_{PN}}, \qquad \frac{A_d}{d_{PN}} = \frac{\mathbf{A}_f\cdot\mathbf{A}_f}{\mathbf{A}_f\cdot\mathbf{PN}}$$

Calculation of Gradients

Calculation of Gradient at Nodal Points

An expression is required to calculate gradient at points *P* and *N* in equation (11.29). One popular method is to use **Gauss**' rule as:

$$\nabla\phi_P = \frac{1}{V}\int_v \nabla\phi dV = \frac{1}{V}\oint_s \phi d\mathbf{A} = \frac{1}{V_P}\sum_{f=1}^{n_f}\phi_f \mathbf{A}_f \qquad (11.31)$$



where ϕ_f is calculated from

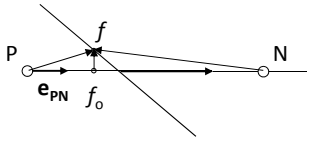
$$\phi_f = \phi_{f_o} + \nabla\phi_{f_o}\cdot\mathbf{f}_o\mathbf{f} \qquad (11.32)$$

where

$$\phi_{f_o} = (1-f_P)\phi_P + f_P\phi_N, \qquad f_P = \frac{\mathbf{Pf}\cdot\mathbf{e}_{PN}}{(\mathbf{PN}\cdot\mathbf{PN})^{1/2}}$$

$$\nabla\phi_{fo} = (1-f_P)\nabla\phi_P + f_P\nabla\phi_N$$

$$\mathbf{f}_o\mathbf{f} = \mathbf{Pf} - \mathbf{Pf}_o = \mathbf{Pf} - (\mathbf{Pf}\cdot\mathbf{e}_{\mathbf{PN}})\mathbf{e}_{\mathbf{PN}}$$



An iterative process is required to calculate gradients:

Step 1: Calculate gradient from Eq. (11.31)

Step 2: Calculate ϕ_f from Eq. (11.32)

Step 3: Repeat steps 1 and 2 until convergence. (4-5 repetitions required)

Diffusion source term

Note that in the general transport equation (11.4), only part of the diffusion term appears explicitly. The other part is buried in the source term:

$$\frac{\partial(\rho\phi)}{\partial t} + div(\rho\phi\mathbf{v}) = div(\Gamma\, grad\,\phi) + s_\phi$$

For example, in the x-momentum equation (2.32a) the diffusion term is

$$\frac{\partial}{\partial x}\left[2\mu\frac{\partial u}{\partial x} + \lambda\, div\,\mathbf{u}\right] + \frac{\partial}{\partial y}\left[\mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)\right] + \frac{\partial}{\partial z}\left[\mu\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial x}\right)\right]$$

$$= \frac{\partial}{\partial x}\left(\mu\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(\mu\frac{\partial u}{\partial y}\right) + \frac{\partial}{\partial z}\left(\mu\frac{\partial u}{\partial z}\right) \rightarrow div(\mu\, grad\, u)$$

$$+ \left[\frac{\partial}{\partial x}\left(\mu\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(\mu\frac{\partial v}{\partial x}\right) + \frac{\partial}{\partial z}\left(\mu\frac{\partial w}{\partial x}\right)\right] + \frac{\partial}{\partial x}(\lambda\, div\,\mathbf{u}) \rightarrow s_{Mx}$$

$$= div(\mu\, grad\, u) + s_{Mx}$$

For incompressible fluids $div\mathbf{u} = 0$. If further, μ = constant, then $s_{Mx} = 0$ (from discussion just after equation (2.34)).

Diffusion source term

However, sometimes μ may be a function of temperature. Morever, in the momentum equations of turbulence models, μ is replaced by $(\mu + \mu_t)$, where μ_t is the turbulent viscosity, which is not constant at all. For those cases, s_{Mx} will not be zero.
The source term s_{Mx} can be expressed in vector form as

$$s_{Mx} = div(\mu grad\mathbf{u}\cdot\mathbf{i}) = div(\mu\nabla^x\mathbf{u})$$

where ∇^x is the x-component of the gradient operator $\nabla = \frac{\partial}{\partial x}\mathbf{i} + \frac{\partial}{\partial y}\mathbf{j} + \frac{\partial}{\partial z}\mathbf{k}$
and $\mathbf{u} = u\mathbf{i} + v\mathbf{j} + w\mathbf{k}$ is the velocity vector.
Integrating over the control volume gives:

$$S_u^{diff} = \int_{CV} div(\mu\nabla^x\mathbf{u})dV = \int_S (\mu\nabla^x\mathbf{u})\cdot\mathbf{n}dA \approx \sum_{f=nb(P)} (\mu\nabla^x u)_f A_f^x + (\mu\nabla^x v)_f A_f^y + (\mu\nabla^x w)_f A_f^z$$

Similarly, the corresponding terms for the y- and z-momentum equations are

$$S_v^{diff} = \int_{CV} div(\mu\nabla^y\mathbf{u})dV = \int_S (\mu\nabla^y\mathbf{u})\cdot\mathbf{n}dA \approx \sum_{f=nb(P)} (\mu\nabla^y u)_f A_f^x + (\mu\nabla^y v)_f A_f^y + (\mu\nabla^y w)_f A_f^z$$

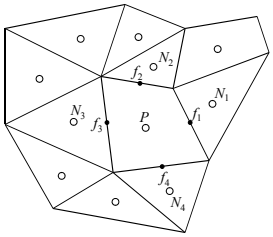
$$S_w^{diff} = \int_{CV} div(\mu\nabla^z\mathbf{u})dV = \int_S (\mu\nabla^z\mathbf{u})\cdot\mathbf{n}dA \approx \sum_{f=nb(P)} (\mu\nabla^z u)_f A_f^x + (\mu\nabla^z v)_f A_f^y + (\mu\nabla^z w)_f A_f^z \quad (11.33)$$

The gradients $\nabla u, \nabla v$ and ∇w at the nodal points are found using Gauss rule (eqn.(11.31) and their face values are found via linear interpolation using eqn. (11.30).

Convection term

- The convection term is approximated as

$$\oint_A (\rho\phi\mathbf{v})\cdot d\mathbf{A} \approx \sum_{f=1}^{n_{fP}} (\rho\mathbf{v}\cdot\mathbf{A})_f\phi_f = \sum_{f=1}^{n_{fP}} \dot{m}_f\phi_f \quad (11.33)$$



n_{fP}: number of faces of cell P, (faces f_1, f_2, f_3, f_4 for cell P in the figure).

$$\dot{m}_f = (\rho\mathbf{v}\cdot\mathbf{A})_f \quad \text{(Mass flow rate through face } f\text{)} \quad (11.34)$$

To calculate m_f, $\mathbf{v}_f$ at the cell faces should be found. $\mathbf{v}_f$ is calculated using the momentum interpolation method (MIM) of Rhie and Chow, to be discussed later.

Defining convective flux as

$$F_f^c = \dot{m}_f\phi_f$$

where ϕ_f is the value of a scalar at the cell face f, the convection term can also be expressed as

$$\oint_A (\rho\phi\mathbf{v})\cdot d\mathbf{A} \approx \sum_{f=1}^{n_{fP}} F_f^c \quad (11.35)$$

Calculation of scalar ϕ_f for Convection Terms

a) First order methods

1) Upwind difference method (UD)

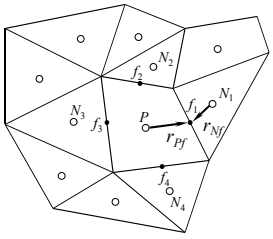
$$\begin{aligned} \phi_f &= \phi_P \quad \text{for} \quad \dot{m}_f > 0 \\ \phi_f &= \phi_N \quad \text{for} \quad \dot{m}_f < 0 \end{aligned} \tag{11.36}$$

The convective flux for the upwind difference method can be written as

$$F_f^U = \max(\dot{m}_f, 0.)\phi_P + \min(\dot{m}_f, 0.)\phi_N \tag{11.37}$$

High Order Methods

1) Linear Upwind Difference Scheme (LUD), or 2nd order Upwind Difference Scheme (SOU)



LUD formulation along a line in 1-D is:

$$\phi_f = \phi_f^U + \left(\frac{\phi_P - \phi_U}{\Delta x} \right) \frac{1}{2} \Delta x$$

where subscript U refers to upwind node. This can be extended to 3D space using Taylor series expansion around point U:

$$\phi_f = \phi_f^U + \nabla \phi_U \cdot \mathbf{r}_U \tag{11.38}$$

where $\phi_f^U = \phi_P, \quad \nabla \phi_U = \nabla \phi_P, \quad \mathbf{r}_U = \mathbf{r}_{Pf} = \mathbf{r}_f - \mathbf{r}_P \quad$ for $\dot{m}_f > 0$

$\phi_f^U = \phi_N, \quad \nabla \phi_U = \nabla \phi_N, \quad \mathbf{r}_U = \mathbf{r}_{Nf} = \mathbf{r}_f - \mathbf{r}_N \quad$ for $\dot{m}_f < 0$

or

$$\begin{aligned} \phi_f &= \phi_P + \nabla \phi_P \cdot \mathbf{r}_{Pf} \quad \text{for} \quad \dot{m}_f > 0 \\ \phi_f &= \phi_N + \nabla \phi_N \cdot \mathbf{r}_{Nf} \quad \text{for} \quad \dot{m}_f < 0 \end{aligned} \tag{11.39}$$

In Eq. (11.39) $\nabla \phi_P$ and $\nabla \phi_N$ are calculated using Eq. (11.31)

Method of Deferred Correction

The convection term $F_f = \dot{m}_f \phi_f$ for **high order methods** at the cell faces is implemented as the sum of the upwind value and other higher order terms which are evaluated at the previous iteration:

$$F_f = F_f^U + (F_f^H - F_f^U)^{old} \tag{11.40}$$

Subscript $U \rightarrow$ refers to upwind (ϕ calculated using first order upwind method)
Subscript H $\rightarrow$ refers to a high order method such as CD or LUD.

Substituting F_f^U from Eqn. (11.37) into Eqn. (11.40)

$$F_f = \max(\dot{m}_f, 0.)\phi_P + \min(\dot{m}_f, 0.)\phi_N + \left[\dot{m}_f \phi_f^H - \max(\dot{m}_f, 0.)\phi_P - \min(\dot{m}_f, 0.)\phi_N\right]^{old}$$

Then, convection term in Eqn. (11.35) becomes

$$\oint_A (\rho\phi\mathbf{v})\cdot d\mathbf{A} \approx \underbrace{\sum_{f=1}^{n_f(P)} \max(\dot{m}_f, 0.)\phi_P}_{implicit} + \underbrace{\sum_{f=1}^{n_f(P)} \min(\dot{m}_f, 0.)\phi_N}_{implicit} + \underbrace{\left[\sum_{f=1}^{n_f(P)} \left(\dot{m}_f \phi_f^H - \max(\dot{m}_f, 0.)\phi_P - \min(\dot{m}_f, 0.)\phi_N\right)\right]^{old}}_{explicit}$$

Source term

Source term in Eq. (11.8) is discretized as

$$\int_{CV} s_\phi dV = s_\phi V_P = (s_c^{eqn} + s_p^{eqn}\phi_P)V_P \tag{11.41}$$

For x-momentum equation: $s_c^{eqn} = -\dfrac{\partial p}{\partial x} = -(\nabla p)_P^x, \quad s_p^{eqn} = 0$

For y-momentum equation: $s_c^{eqn} = -\dfrac{\partial p}{\partial y} = -(\nabla p)_P^y, \quad s_p^{eqn} = 0$

where $\nabla p = \dfrac{\partial p}{\partial x} i + \dfrac{\partial p}{\partial y} j = (\nabla p)^x \mathbf{i} + (\nabla p)^y \mathbf{j}$

Using Gauss' rule (Eq. (11.31 with (11.32)):

$$\nabla p_P = \frac{1}{V}\int_v \nabla p dV = \frac{1}{V}\oint_s p d\mathbf{A} = \frac{1}{V_P}\sum_{f=1}^{n_f} p_f \mathbf{A}_f \quad \text{(Coservative form)} \tag{11.42}$$

$$\nabla p_P = \frac{1}{V_P}\left(\sum_{f=1}^{n_f} p_f \mathbf{A}_f^x\right)\mathbf{i} + \frac{1}{V_P}\left(\sum_{f=1}^{n_f} p_f \mathbf{A}_f^y\right)\mathbf{j} = (\nabla p)^x \mathbf{i} + (\nabla p)^y \mathbf{j} \quad \text{(Noncoservative form)}$$

$$\int_{CV} s_\phi dV = s_\phi V_P = -\frac{\partial p}{\partial x} V_P = -(\nabla p)_P^x V_P = -\sum_{f=1}^{n_f} p_f \mathbf{A}_f^x \quad \text{for x-momentum equation} \tag{11.43}$$

$$\int_{CV} s_\phi dV = s_\phi V_P = \frac{\partial p}{\partial y} V_P = -(\nabla p)_P^y V_P = -\sum_{f=1}^{n_f} p_f \mathbf{A}_f^y \quad \text{for y-momentum equation}$$

Substituting the discretized forms of the transient, convection and diffusion terms into the general equation (11.8)

$$\frac{(\rho\phi V)_P-(\rho\phi V)_P^0}{\Delta t}+\sum_{f=1}^{n_f(P)}\max(\dot m_f,0.)\phi_P+\sum_{f=1}^{n_f(P)}\min(\dot m_f,0.)\phi_N=\sum_{f=1}^{n_f(P)}\frac{\Gamma_F A_d}{d_{PN}}(\phi_N-\phi_P)$$
$$+\sum_{f=1}^{n_f(P)}\left\{\Gamma_f\left(\overline{\nabla\phi}\right)_f.\mathbf{A}_t+\Gamma_f\frac{(\nabla\phi_N-\nabla\phi_P)\cdot\mathbf{PP'}}{d_{PN}}A_d\right\}^{n-1}+s_\phi V_P$$
$$-\sum_{f=1}^{n_f(P)}\left[\dot m_f\phi_f^H-\max(\dot m_f,0.)\phi_P-\min(\dot m_f,0.)\phi_N\right]^{n-1} \qquad (11.44)$$

Equation (11.44) can be written in the form of

$$a_P\phi_P=\sum_{N=nb(P)}a_N\phi_N+S \qquad (11.45)$$

$S=s_c^{eqn}V_P+S^{trans}+S^{dc}+S^{pres}=$ (source in differential equation + transient + deferred correction + pressure) terms

$$a_N=\frac{\Gamma_f A_d}{d_{PN}}-\min\left[\dot m_f,\,0\right],\qquad a_P=\sum_{N=nb(P)}a_N+\frac{\rho_P V_P}{\Delta t}-s_p^{eqn}V_P,$$

$$S^{trans}=a_P^o\phi_P^o,\qquad a_P^o=\frac{\rho_P^0 V_P}{\Delta t}$$

$$S^{dc}=-\sum_{f=1}^{n_f(P)}\left[\dot m_f\phi_f^H-\max(\dot m_f,0.)\phi_P-\min(\dot m_f,0.)\phi_N\right]^{n-1}$$
$$+\sum_{f=1}^{n_f(P)}\left\{\Gamma_f\left(\overline{\nabla\phi}\right)_f.\mathbf{A}_t+\Gamma_f\frac{(\nabla\phi_N-\nabla\phi_P)\cdot\mathbf{PP'}}{d_{PN}}A_d\right\}^{n-1}, \qquad (11.46)$$

$$S^{pres}=\begin{cases}=-(\nabla p)_P^x V_P=-\sum_{f=1}^{n_f}p_f\mathbf{A}_f^x & \text{for } x\text{-momentum equation}\\ =-(\nabla p)_P^y V_P=-\sum_{f=1}^{n_f}p_f\mathbf{A}_f^y & \text{for } y\text{-momentum equation}\end{cases}$$

$$\mathbf{PP'}=\left|\mathbf{f_1f}\right|\mathbf{e_t},\ \left|\mathbf{f_1f}\right|=\left|\mathbf{f_of}\right|\frac{A_d}{A_f},\ \mathbf{f_of}=\mathbf{Pf}-\mathbf{Pf}_o=\mathbf{Pf}-(\mathbf{Pf}\cdot\mathbf{e_{PN}})\mathbf{e_{PN}},\ \frac{A_d}{d_{PN}}=\frac{\mathbf{A}_f\cdot\mathbf{A}_f}{\mathbf{A}_f\cdot\mathbf{PN}},\quad \mathbf{A}_t=\mathbf{A}_f-A_d\mathbf{e}_{PN},\ A_d=\left|\mathbf{A}_d\right|=\frac{\mathbf{A}_f\cdot\mathbf{A}_f}{\mathbf{A}_f\cdot\mathbf{e}_{PN}}$$

$s^{eqn}=s_c^{eqn}+s_p^{eqn}\phi=$ source terms per unit volume in the differential equation

= body forces per unit volume in momentum equations

= energy generation rate per unit volume in energy equation

$S=sV$

For a degree of implicitness θ, Equation (11.44) can be written in the form of

$$a_P \phi_P = \theta \sum_{N=nb(P)} a_N \phi_N + S \qquad \theta = 1 \text{ for fully implicit, } \theta = 0.5 \text{ for Crank-Nicolson, } \theta = 0 \text{ for explicit scheme}$$

$S = s_c^{eqn} V_P + S^{trans} + S^{dc} + S^{pres} + S^{imp} =$ (source in d.e. + transient + deferred correction + pressure + degree of implicitness) terms

$$a_N = \frac{\Gamma_f A_d}{d_{PN}} - \min\left[\dot{m}_f, 0\right], \qquad a_P = \theta \sum_{N=nb(P)} a_N + \frac{\rho_P V_P}{\Delta t} - \theta s_p^{eqn} V_P,$$

$$S^{trans} = a_P^o \phi_P^o, \qquad a_P^o = \frac{\rho_P^0 V_P}{\Delta t}$$

$$S^{dc} = -\sum_{f=1}^{n_f(P)} \left[\dot{m}_f \phi_f^H - \max(\dot{m}_f, 0.)\phi_P - \min(\dot{m}_f, 0.)\phi_N\right]^{n-1} + \sum_{f=1}^{n_f(P)} \left\{\Gamma_f \left(\overline{\nabla\phi}\right)_f \cdot \mathbf{A}_t + \Gamma_f \frac{(\nabla\phi_N - \nabla\phi_P)\cdot \mathbf{PP}'}{d_{PN}} A_d\right\}^{n-1},$$

$$S^{pres} = \begin{cases} = -(\nabla p)_P^x V_P = -\sum_{f=1}^{n_f} p_f \mathbf{A}_f^x & \text{for } x\text{-momentum equation} \\ = -(\nabla p)_P^y V_P = -\sum_{f=1}^{n_f} p_f \mathbf{A}_f^y & \text{for } y\text{-momentum equation} \end{cases}$$

$$S^{imp} = (1-\theta) \sum_{N=nb(P)} a_N (\phi_N^0 - \phi_P^0) + (1-\theta) s_P^{eqn} \phi_P^0$$

$s^{eqn} = s_c^{eqn} + s_p^{eqn}\phi =$ source terms per unit volume in the differential equation

= body forces per unit volume in momentum equations

= energy generation rate per unit volume in energy equation

$S = sV$

In above equations superscripts:

n–1 → refers to previous iteration values

o → refers to previous time step values

In solving steady flows, Eq. (11.45) is relaxed as

$$\frac{a_P}{\alpha}\phi_P = \sum_{N=nb(P)} a_N \phi_N + S + (1-\alpha)\frac{a_P}{\alpha}\phi_P^{n-1} \qquad (11.47)$$

$0 \le \alpha \le 1$ underrelaxation factor.

In that case, the terms in Eq. (11.46) are redefined as

$$\begin{cases} S \leftarrow S + (1-\alpha)\dfrac{a_P}{\alpha}\phi_P^{n-1} \\ a_P \leftarrow \dfrac{a_P}{\alpha} \end{cases} \qquad (11.48)$$

where superscript n–1 refers to the previous iteration value.

MIM method

To find m_f in Eq.(11.34), velocity vector $\mathbf{v}_f$ at cell faces are calculated using the momentum interpolation method (MIM) originally proposed by Rhie and Chow. Rewriting the momentum equations for a 2D flow in the form of Eq. (11.45)

$$\frac{a_P^u}{\alpha}u_P = \sum_{N=nb(P)} a_N^u u_N + b_P^u - V(\nabla p)_P.\mathbf{i} \tag{11.49}$$

$$\frac{a_P^v}{\alpha}v_P = \sum_{N=nb(P)} a_N^v v_N + b_P^v - V(\nabla p)_P.\mathbf{j} \tag{11.50}$$

Where, b is the source terms excluding pressure source. Above Eqns can be written as,

$$\begin{Bmatrix} u_P \\ v_P \end{Bmatrix} - \begin{Bmatrix} H[u]_P \\ H[v]_P \end{Bmatrix} = -\begin{bmatrix} D[u]_P & 0 \\ 0 & D[v]_P \end{bmatrix} \begin{Bmatrix} (\nabla p)_P \cdot \mathbf{i} \\ (\nabla p)_P \cdot \mathbf{j} \end{Bmatrix} \tag{11.51}$$

where

$$H[\phi]_P = \frac{\sum_{N=nb(P)} a_N^\phi \phi_N + b_P^\phi}{a_P^\phi / \alpha} \qquad D[\phi]_P = \frac{V}{a_P^\phi / \alpha} \tag{11.52}$$

Defining the vector forms of the above operators as

$$\mathbf{H}[\mathbf{v}]_P = \begin{bmatrix} H[u]_P \\ H[v]_P \end{bmatrix} \quad \mathbf{D}_P = \begin{bmatrix} D[u]_P & 0 \\ 0 & D[v]_P \end{bmatrix} \quad (\nabla P)_P = \begin{bmatrix} (\nabla p)_P \cdot \mathbf{i} \\ (\nabla p)_P \cdot \mathbf{j} \end{bmatrix} = \begin{bmatrix} (\nabla P)_P^x \\ (\nabla P)_P^y \end{bmatrix} \tag{11.53}$$

For point P, the momentum equation in vector form becomes

$$\mathbf{v}_P = \mathbf{H}[\mathbf{v}]_P - \mathbf{D}_P(\nabla p)_P \tag{11.54}$$

MIM method

Similarly, for point N

$$\mathbf{v}_N = \mathbf{H}[\mathbf{v}]_N - \mathbf{D}_N(\nabla p)_N$$

Eq. (11.54) can be written at the cell face f as

$$\mathbf{v}_f = \mathbf{H}_f - \mathbf{D}_f(\nabla p)_f \tag{11.55}$$

Assuming that the term $\mathbf{H}_f$ can be approximated by using an interpolation between points P and N, that is

$$\mathbf{H}_f = \overline{\mathbf{H}_f}$$

where the overbar indicates linear average between point P and N. Then, Eq. (11.55) becomes

$$\mathbf{v}_f = \overline{\mathbf{H}_f} - \mathbf{D}_f(\nabla p)_f \tag{11.56}$$

The linearly averaged form of Eq.(11.55) is

$$\overline{\mathbf{H}_f} = \overline{\mathbf{v}_f} + \overline{\mathbf{D}_f \nabla p_f} \tag{11.57}$$

Using, $\overline{\mathbf{D}_f \nabla p_f} = \overline{\mathbf{D}}_f \overline{\nabla p_f}$

Eq (11.57) becomes, $\overline{\mathbf{H}_f} = \overline{\mathbf{v}_f} + \overline{\mathbf{D}}_f \overline{\nabla p_f}$ (11.58)

Mass flow rate

where $\overline{\mathbf{v}_f} = (1-f_P)\mathbf{v}_P + f_P\mathbf{v}_N$

$\overline{\mathbf{D}_f} = (1-f_P)\mathbf{D}_P + f_P\mathbf{D}_N$

$\overline{\nabla p_f} = (1-f_P)(\nabla p)_P + f_P(\nabla p)_N$

$\mathbf{D}_f = \overline{\mathbf{D}_f}$

where, f_P is the geometric interpolation factor defined as

$$f_P = \frac{\overline{Pf}}{\overline{PN}} \quad \text{or preferably:} \quad f_P = \frac{\mathbf{Pf}\cdot\mathbf{e}_{PN}}{\overline{PN}} = \frac{\mathbf{Pf}\cdot\mathbf{e}_{PN}}{(\mathbf{PN}\cdot\mathbf{PN})^{1/2}}$$

A more accurate interpolated value can be found using Eqn. (11.32).

Substituting eq (11.58) into eq (11.56)

$$\mathbf{v}_f = \overline{\mathbf{v}_f} - \left(\mathbf{D}_f(\nabla p)_f - \overline{\mathbf{D}}_f\overline{\nabla p_f}\right) \tag{11.59}$$

Mass flow rate: $\dot{m}_f = (\rho\mathbf{v}\cdot\mathbf{A})_f$

Substituting Eq. (11.59) for $\mathbf{v}_f$:

$$\dot{m}_f = \rho_f\overline{\mathbf{v}_f}\cdot\mathbf{A}_f - \left(\rho_f\mathbf{D}_f\nabla p_f\cdot\mathbf{A}_f - \rho_f\overline{\mathbf{D}}_f\overline{(\nabla p)_f}\cdot\mathbf{A}_f\right) \tag{11.60}$$

Mass flow rate

The term $\nabla p_f.\mathbf{A}_f$ is discretized using Eq. (11.34)

$$\nabla p_f.\mathbf{A}_f = \frac{p_N - p_P}{d_{PN}}A_d + \overline{\nabla p_f}.\mathbf{A}_t + \frac{(\nabla p_N - \nabla p_P)\cdot\mathbf{PP'}}{d_{PN}}A_d \tag{11.61}$$

where the overbar denotes linear interpolation between the two cells straddling the surface *f*.

Substituting into Eq. (11.60)

$$\dot{m}_f = \rho_f\overline{\mathbf{v}_f}\cdot\mathbf{A}_f + \rho_f\overline{\mathbf{D}}_f\overline{(\nabla p)_f}\cdot\mathbf{A}_f - \rho_f\mathbf{D}_f\left(\frac{p_N - p_P}{d_{PN}}A_d + \left(\overline{\nabla p}\right)_f.(\mathbf{A}_t)_f + \frac{(\nabla P_N - \nabla P_P)\cdot\mathbf{PP'}}{d_{PN}}A_d\right) \tag{11.62}$$

or using $\frac{A_d}{d_{PN}} = \frac{\mathbf{A}_f\cdot\mathbf{A}_f}{\mathbf{A}_f\cdot\mathbf{PN}}$

$$\dot{m}_f = \rho_f\overline{\mathbf{v}_f}\cdot\mathbf{A}_f + \rho_f\overline{\mathbf{D}}_f\overline{(\nabla p)_f}\cdot\mathbf{A}_f - \rho_f\frac{(\mathbf{D}_f\mathbf{A}_f)\cdot\mathbf{A}_f}{\mathbf{A}_f\cdot\mathbf{PN}}(p_N - p_P) - \rho_f(\mathbf{D}_f\overline{\nabla p_f})\cdot(\mathbf{A}_t)_f - \rho_f\frac{(\mathbf{D}_f\mathbf{A}_f)\cdot\mathbf{A}_f}{\mathbf{A}_f\cdot\mathbf{PN}}(\nabla p_N - \nabla p_P)\cdot\mathbf{PP'} \tag{11.63}$$

$\rho_f\frac{(\mathbf{D}_f\mathbf{A}_f)\cdot\mathbf{A}_f}{\mathbf{A}_f\cdot\mathbf{PN}} = a_N^{p'}$ from Eq. (11.77)

SIMPLE Algorithm

Consider the momentum equations in Eq. (11.56) rewritten at the face f as

$$\mathbf{v}_f - \mathbf{H}[\mathbf{v}]_f = -\mathbf{D}_f(\nabla p)_f \tag{11.64}$$

Let p', u', v' be the correction needed to correct the guessed pressure and velocity fields, i.e.

$$p = p^* + p' \quad \text{and} \quad \mathbf{v} = \mathbf{v}^* + \mathbf{v}' \tag{11.65}$$

For a guessed pressure field p^* the corresponding velocity can be written as

$$\mathbf{v}_f^* - \mathbf{H}[\mathbf{v}^*]_f = -\mathbf{D}_f(\nabla p^*)_f \tag{11.66}$$

Subtracting Eq. (11.66) from Eq. (11.64) yields

$$\mathbf{v}'_f - \mathbf{H}[\mathbf{v}']_f = -\mathbf{D}_f(\nabla p')_f \tag{11.67}$$

SIMPLE Algorithm

In SIMPLE method the second term on the left hand side of Eq. (11.67) is neglected. Then, Eq. (11.67) becomes

$$\mathbf{v}'_f = -\mathbf{D}_f \nabla p'_f \tag{11.68}$$

The discretized continuity equation is

$$\frac{(\rho_P - \rho_P^o)}{\Delta t} V_P + \sum_{f=nb(P)} \dot{m}_f = 0 \quad \text{where} \quad \dot{m}_f = \rho \mathbf{v}_f \cdot \mathbf{A}_f \tag{11.69}$$

or

$$\frac{(\rho_P - \rho_P^o)}{\Delta t} V_P + \sum_{f=nb(P)} \dot{m}_f^* + \sum_{f=nb(P)} \dot{m}'_f = 0 \tag{11.70}$$

where $\dot{m}'_f = (\rho \mathbf{v}')_f \cdot \mathbf{A}_f$

Substituting $\mathbf{v}'_f$ from Eq. (11.68)

$$\dot{m}'_f = (\rho \mathbf{v}')_f \cdot \mathbf{A}_f = -\rho_f (\mathbf{D}_f \nabla p'_f) \cdot \mathbf{A}_f \tag{11.71}$$

SIMPLE Algorithm

The term $\nabla p'_f.\mathbf{A}_f$ is discretized using Eq. (11.34)

$$\nabla p'_f.\mathbf{A}_f = \frac{p'_N - p'_P}{d_{PN}} A_d + \overline{\nabla p'_f}.\mathbf{A}_t + \frac{(\nabla p'_N - \nabla p'_P)\cdot \mathbf{PP'}}{d_{PN}} A_d \qquad (11.72)$$

Substituting Eq. (11.72) into Eq. (11.71) gives

$$\dot{m}'_f = -\rho_f \frac{(\mathbf{D}_f\mathbf{A}_f)\cdot\mathbf{A}_f}{\mathbf{A}_f\cdot\mathbf{PN}}(p'_N - p'_P) - \rho_f \mathbf{D}_f \left(\overline{\nabla p'}\right)_f .(\mathbf{A}_t)_f - \rho_f \frac{(\mathbf{D}_f\mathbf{A}_f)\cdot\mathbf{A}_f}{\mathbf{A}_f\cdot\mathbf{PN}}(\nabla p'_N - \nabla p'_P)\cdot\mathbf{PP'} \qquad (11.73)$$

Substituting Eq. (11.73) into Eq. (11.70) , the continuity equation becomes

$$\sum_{f=nb(P)} \rho_f \frac{(\mathbf{D}_f\mathbf{A}_f)\cdot\mathbf{A}_f}{\mathbf{A}_f\cdot\mathbf{PN}}(p'_N - p'_P) = \sum_{f=nb(P)} \dot{m}^*_f + \frac{(\rho_P - \rho^o_P)}{\Delta t} V_P - \sum_{f=nb(P)} \rho_f \mathbf{D}_f \left(\overline{\nabla p'}\right)_f .(\mathbf{A}_t)_f \qquad (11.74)$$
$$- \sum_{f=nb(P)} \rho_f \frac{(\mathbf{D}_f\mathbf{A}_f)\cdot\mathbf{A}_f}{\mathbf{A}_f\cdot\mathbf{PN}}(\nabla p'_N - \nabla p'_P)\cdot\mathbf{PP'}$$

Equation (11.74) is also called the pressure correction equation.

SIMPLE Algorithm

The pressure correction equation (11.74) can be written as:

$$a^{p'}_P p'_P = \sum_{N=nb(P)} a^{p'}_N p'_N + b^{p'}_P \qquad (11.77)$$

$$a^{p'}_N = \rho_f \frac{(\mathbf{D}_f\mathbf{A}_f)\cdot\mathbf{A}_f}{\mathbf{A}_f\cdot\mathbf{PN}}$$

$$a^{p'}_P = \sum_{N=nb(P)} a^{p'}_N$$

$$b^{p'}_P = -\frac{(\rho_P - \rho^o_P)}{\Delta t} V_P - \sum_{f=nb(P)} \dot{m}^*_f + \sum_{f=nb(P)} \rho_f \mathbf{D}_f \overline{\nabla p'_f}\cdot(\mathbf{A}_t)_f$$
$$+ \sum_{f=nb(P)} \rho_f \frac{(\mathbf{D}_f\mathbf{A}_f)\cdot\mathbf{A}_f}{\mathbf{A}_f\cdot\mathbf{PN}}(\nabla p'_N - \nabla p'_P)\cdot\mathbf{PP'}$$

The last term on the right hand side of b_P term can be neglected.

$$\mathbf{D}_f = \begin{pmatrix} D^u_f & 0 \\ 0 & D^v_f \end{pmatrix} \qquad (11.78)$$

SIMPLE Algorithm

From Eq. (11.73), the mass correction is

$$\dot{m}'_f = -a_N^{p'}(p'_N - p'_P) - \rho_f \mathbf{D}_f \left(\overline{\nabla p'}\right)_f .(\mathbf{A}_t)_f - a_N^{p'}(\nabla p'_N - \nabla p'_P)\cdot \mathbf{PP}' \qquad (11.79)$$

After solving the p' field from Eq. (11.77) the mass flow rate at the surfaces are corrected by using Eq. (11.79)

$$\dot{m}_f = \dot{m}_f^* - a_N^{p'}(p'_N - p'_P) - \rho_f \mathbf{D}_f \overline{\nabla p'_f}\cdot(\mathbf{A}_t)_f - a_N^{p'}(\nabla p'_N - \nabla p'_P)\cdot \mathbf{PP}' \qquad (11.80)$$

The nodal velocities are corrected as

$$\mathbf{v}_P = \mathbf{v}_P^* + \mathbf{v}'_P = \mathbf{v}_P^* - \mathbf{D}_P \nabla p'_P \qquad (11.81)$$

and the pressure field is corrected by using

$$p_P = p_P^* + \alpha_p p'_P \qquad (11.82)$$

α_p = pressure under-relaxation factor

SIMPLE Algorithm

- Step 1: Solve the discretized transport equation (11.47) for $\phi = u$ and $\phi = v$

$$\frac{a_P}{\alpha}\phi_P = \sum_{N=nb(P)} a_N \phi_N + S + (1-\alpha)\frac{a_P}{\alpha}\phi_P^{n-1}$$

- Step 2: Calculate mass flow rate at cell faces (Eqn. (11.63))

$$\dot{m}_f = \rho_f \overline{\mathbf{v}_f}\cdot\mathbf{A}_f + \rho_f \overline{\mathbf{D}}_f \overline{(\nabla p)_f}\cdot\mathbf{A}_f - a_N^{p'}(p_N - p_P) - \rho_f(\mathbf{D}_f \overline{\nabla p_f})\cdot(\mathbf{A}_t)_f - a_N^{p'}(\nabla p_N - \nabla p_P)\cdot\mathbf{PP}'$$

- Step3: Solve pressure correction equation (11.77)

$$a_P^{p'} p'_P = \sum_{N=nb(P)} a_N^{p'} p'_N + b_P^{p'}$$

- Step 4: Correct velocities and pressure at points P using Eqn's (11.81), (11.82)

$$\mathbf{v}_P = \mathbf{v}_P^* + \mathbf{v}'_P = \mathbf{v}_P^* - \mathbf{D}_P \nabla p'_P \qquad p_P = p_P^* + \alpha_p p'_P$$

- Step 5: Correct mass flow rate using equation (11.80) :

$$\dot{m}_f = \dot{m}_f^* - a_N^{p'}(p'_N - p'_P) - \rho_f \mathbf{D}_f \overline{\nabla p'_f}\cdot(\mathbf{A}_t)_f - a_N^{p'}(\nabla p'_N - \nabla p'_P)\cdot\mathbf{PP}'$$

- Step 6: Solve the discretized transport equation (11.47) for other unknowns (i.e. for ϕ = Temperature)

$$\frac{a_P}{\alpha}\phi_P = \sum_{N=nb(P)} a_N \phi_N + S + (1-\alpha)\frac{a_P}{\alpha}\phi_P^{n-1}$$

- Step 7: Repeat steps 1 to 6 until convergence.

Modifications to original MIM method

In the original MIM method proposed by Rhie and Chow, using equations (11.56-11.57) for the face values $\mathbf{H}_f$ and $\mathbf{D}_f$, makes the solution to depend on the relaxation parameter α as well as the time step size Δt. To overcome this, Eqns (11.49-11.50) are written as (Bo Yu et al. , *Numer. Heat Transf.*, Part B, v42, pp. 141-166, 2002) :

$$\mathbf{v}_P = \mathbf{H}_P - \frac{\alpha}{\mathbf{a}_P} V_P (\nabla p)_P + (1-\alpha)\mathbf{v}_P^{n-1} + \frac{\alpha}{\mathbf{a}_P} a_P^o \mathbf{v}_P^o + \frac{\alpha}{\mathbf{a}_P} V_P (\mathbf{s}^{body})_P$$

where

$$\mathbf{H}_P = \alpha \mathbf{a}_P^{-1} \left[\left(\sum_{N=nb(P)} \mathbf{a}_N \mathbf{v}_N \right) + (\mathbf{S}^{bc})_P + (\mathbf{S}^{dc})_P \right] \qquad \text{(a)}$$

Note that the source terms resulting from relaxation, unsteady term and body forces has been separated from **H** term. $\mathbf{S}^{bc}$ is the source due to boundary conditions.
The corresponding equation at the face is

$$\mathbf{v}_f = \mathbf{H}_f - \frac{\alpha}{\mathbf{a}_f} V_f (\nabla p)_f + (1-\alpha)\mathbf{v}_f^{n-1} + \frac{\alpha}{\mathbf{a}_f} (a_P^o)_f \mathbf{v}_f^o + \frac{\alpha}{\mathbf{a}_f} V_f (\mathbf{s}^{body})_f \qquad \text{(b)}$$

or

$$\mathbf{v}_f = \mathbf{H}_f - \mathbf{D}_f (\nabla p)_f + (1-\alpha)\mathbf{v}_f^{n-1} + \frac{\alpha}{\mathbf{a}_f} (a_P^o)_f \mathbf{v}_f^o + \mathbf{D}_f (\mathbf{s}^{body})_f \qquad \text{(c)}$$

Modifications to original MIM method

where

$$\overline{\mathbf{H}_f} = (1-f_P)\mathbf{H}_P + f_P \mathbf{H}_N$$

Following the procedure as in Rhie and Chow method, the face velocity can be expressed as

$$\begin{aligned} \mathbf{v}_f = \overline{\mathbf{v}_f} &- \left[\mathbf{D}_f (\nabla p)_f - \overline{\mathbf{D}}_f \overline{(\nabla p)_f} \right] \\ &+ (1-\alpha)\left[\mathbf{v}_f^{n-1} - \overline{\mathbf{v}_f^{n-1}} \right] \\ &+ \alpha (\mathbf{a}_f)^{-1} \left[(a_P^o)_f \mathbf{v}_f^o - \overline{(a_P^o)_f}\, \overline{\mathbf{v}_f^o} \right] \\ &+ \left[\mathbf{D}_f (\mathbf{s}^{body})_f - \overline{\mathbf{D}_f (\mathbf{s}^{body})_f} \right] \end{aligned} \qquad \text{(d)}$$

where

$$\frac{1}{\mathbf{a}_f} = (1-f_P)\frac{1}{\mathbf{a}_P} + f_P \frac{1}{\mathbf{a}_N}$$

$$a_P^o = \frac{\rho V_P}{\Delta t}$$

Modifications to original MIM method

Then, using Eqn. (d), mass flow rate through face f can be written as

$$\begin{aligned}\dot{m}_f = \dot{m}_f^{RC} &+ (1-\alpha)\left[\dot{m}_f^{n-1} - \rho_f \overline{\mathbf{v}_f^{n-1}} \cdot \mathbf{A}_f\right] \\ &+ \frac{\rho_f}{\Delta t}\frac{(\mathbf{D}_f \mathbf{A}_f)\cdot \mathbf{A}_f}{\mathbf{A}_f \cdot \mathbf{A}_f}\dot{m}_f^o - \left(\rho_f \alpha(\mathbf{a}_f)^{-1}\overline{(a_P^o)}_f\,\overline{\mathbf{v}_f^o}\right)\cdot \mathbf{A}_f \\ &+ \rho_f\left[\mathbf{D}_f(\mathbf{s}^{body})_f - \overline{\mathbf{D}_f(\mathbf{s}^{body})_f}\right]\cdot \mathbf{A}_f\end{aligned} \qquad \text{(e)}$$

where

$\dot{m}_f^{RC}$ = mass flow rate calculated from Eqn. (11.63) using Rhie-Chow method.

$\dot{m}_f^o$ = mass flow rate at face f during previous time step.

$\dot{m}_f^{n-1}$ = mass flow rate at face f during previous iteration step.

Note that $\dot{m}_f^{n-1}$ is used during the current iteration so that no extra storage is needed for $\dot{m}_f^{n-1}$. However, storage is needed for $\dot{m}_f^o$.

Modifications to original MIM method

The linearly averaged value of $(\mathbf{s}^{body})_f$ is estimated using

$$\overline{(\mathbf{s}^{body})_f} = (1-f_P)(\mathbf{s}^{body})_P + f_P(\mathbf{s}^{body})_N \qquad \text{(f)}$$

and the face value is estimated as:

$$(\mathbf{s}^{body})_f = \frac{1}{2}((\mathbf{s}^{body})_P + (\mathbf{s}^{body})_N) + \frac{1}{2}(\nabla(\mathbf{s}^{body})_P \cdot \mathbf{r}_{Pf} + \nabla(\mathbf{s}^{body})_N \cdot \mathbf{r}_{Nf}) \qquad \text{(g)}$$

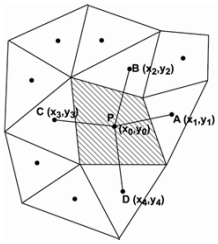
where the gradient at the nodal point P can be calculated using the Gauss' rule.

The use of the neighboring cell source values for the estimation of $(s_c)_f$, is expected to circumvent the convergence difficulties encountered in flows with large source terms.

It should be noted that Eqn. (d) is equivalent to Eqn. (c). However, Eqn (d) may be preferred since extra storage is required for $\mathbf{H}_f$ in Eqn (c).

Gradient reconstruction using Least-Squares method

- One popular method for calculating the gradient at points *P* and *N* is to use **Gauss**' method given by Eqn's (11.31-11.32). Another possible way is to use **least-squares** method.
- Consider a control volume and its neighbors:



- In LUD scheme for the case $\dot{m}_f > 0$, the value of ϕ at a face f can be calculated from Eqn. (11.39) as:

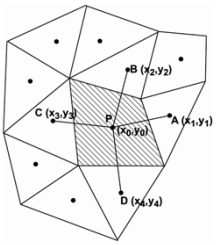
$$\phi_f = \phi_P + \nabla\phi_P \cdot \mathbf{r}_{Pf}$$

- A similar equation can be written between P and its neighbor points A, B, C and D:

or

$$\phi_i = \phi_0 + \left(\frac{\partial\phi}{\partial x}\right)_0 (x_i - x_0) + \left(\frac{\partial\phi}{\partial y}\right)_0 (y_i - y_0)$$

$$\phi_i = \phi_0 + \left(\frac{\partial\phi}{\partial x}\right)_0 \Delta x_i + \left(\frac{\partial\phi}{\partial y}\right)_0 \Delta y_i$$



where $i \rightarrow$ 1, 2, 3, 4,…, or A, B, C, D,...

- For each node around "0" (0 →P) we have

$$\phi_1 - \phi_0 = \left(\frac{\partial\phi}{\partial x}\right)_0 \Delta x_1 + \left(\frac{\partial\phi}{\partial y}\right)_0 \Delta y_1$$

$$\phi_2 - \phi_0 = \left(\frac{\partial\phi}{\partial x}\right)_0 \Delta x_2 + \left(\frac{\partial\phi}{\partial y}\right)_0 \Delta y_2$$

$$\phi_3 - \phi_0 = \left(\frac{\partial\phi}{\partial x}\right)_0 \Delta x_3 + \left(\frac{\partial\phi}{\partial y}\right)_0 \Delta y_3$$

$$\phi_N - \phi_0 = \left(\frac{\partial\phi}{\partial x}\right)_0 \Delta x_N + \left(\frac{\partial\phi}{\partial y}\right)_0 \Delta y_N$$

- This set of equations can be written in matrix form as

$$\begin{bmatrix} \Delta x_1 & \Delta y_1 \\ \Delta x_2 & \Delta y_2 \\ \Delta x_3 & \Delta y_3 \\ : & : \\ \Delta x_N & \Delta y_N \end{bmatrix} \begin{bmatrix} \left(\frac{\partial \phi}{\partial x}\right)_0 \\ \left(\frac{\partial \phi}{\partial y}\right)_0 \end{bmatrix} = \begin{bmatrix} \phi_1 - \phi_0 \\ \phi_2 - \phi_0 \\ \phi_3 - \phi_0 \\ : \\ \phi_N - \phi_0 \end{bmatrix}$$

- This represents an overdetermined system of equations, in the form

$$\mathbf{AX} = \mathbf{B}$$

which may be solved for $\mathbf{X}=[(\partial\phi/\partial x)_0, (\partial\phi/\partial y)_0]$ using least-squares approach. Multiplying both sides of the eqn by the transpose $\mathbf{A^T}$

$$\mathbf{A^T} = \begin{bmatrix} \Delta x_1 & \Delta x_2 & \Delta x_3 & .. & \Delta x_N \\ \Delta y_1 & \Delta y_2 & \Delta y_3 & .. & \Delta y_N \end{bmatrix}$$

- we obtain

$$\mathbf{A^T A X} = \mathbf{A^T B}$$

or

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}}_{\mathbf{A^T A}} \underbrace{\begin{bmatrix} \left(\frac{\partial \phi}{\partial x}\right)_0 \\ \left(\frac{\partial \phi}{\partial y}\right)_0 \end{bmatrix}}_{\mathbf{X}} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \end{bmatrix}}_{\mathbf{A^T B}}$$

where

$$a_{11} = (\Delta x_1)^2 + (\Delta x_2)^2 + (\Delta x_3)^2 + \ldots + (\Delta x_N)^2$$
$$a_{12} = a_{21} = (\Delta x_1)(\Delta y_1) + (\Delta x_2)(\Delta y_2) + (\Delta x_3)(\Delta y_3) + \ldots + (\Delta x_N)(\Delta y_N)$$
$$a_{22} = (\Delta y_1)^2 + (\Delta y_2)^2 + (\Delta y_3)^2 + \ldots + (\Delta y_N)^2$$
$$b_1 = (\Delta x_1)(\phi_1 - \phi_0) + (\Delta x_2)(\phi_2 - \phi_0) + (\Delta x_3)(\phi_3 - \phi_0) + \ldots + (\Delta x_N)(\phi_N - \phi_0)$$
$$b_2 = (\Delta y_1)(\phi_1 - \phi_0) + (\Delta y_2)(\phi_2 - \phi_0) + (\Delta y_3)(\phi_3 - \phi_0) + \ldots + (\Delta y_N)(\phi_N - \phi_0)$$

- It can be observed that the matrix $\mathbf{A^T A}$ is a 2×2 matrix which can be easily inverted to solve for $\mathbf{X}$, i.e.

$$\mathbf{X} = (\mathbf{A^T A})^{-1} \mathbf{A^T B}$$

- For 3D problems, $\mathbf{X}=[(\partial\phi/\partial x)_0, (\partial\phi/\partial y)_0, (\partial\phi/\partial z)_0]$ and $\mathbf{A^T A}$ is a 3×3 matrix.

- Performing the inversion we obtain

$$(\nabla\phi)_0 = \begin{bmatrix} (\nabla^x\phi)_0 \\ (\nabla^y\phi)_0 \\ (\nabla^z\phi)_0 \end{bmatrix} \begin{bmatrix} \left(\frac{\partial\phi}{\partial x}\right)_0 \\ \left(\frac{\partial\phi}{\partial y}\right)_0 \\ \left(\frac{\partial\phi}{\partial z}\right)_0 \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{NB(0)} C_j^x(\phi_j - \phi_0) \\ \sum_{j=1}^{NB(0)} C_j^y(\phi_j - \phi_0) \\ \sum_{j=1}^{NB(0)} C_j^z(\phi_j - \phi_0) \end{bmatrix}$$

where the C terms are

$$C_j^x = \alpha_{j,1} - \frac{r_{12}}{r_{11}}\alpha_{j,2} + \psi\alpha_{j,3}$$

$$C_j^y = \alpha_{j,2} - \frac{r_{23}}{r_{22}}\alpha_{j,3}$$

$$C_j^z = \alpha_{j,3}$$

and

$$\alpha_{j,1} = \frac{\Delta x_j}{r_{11}^2}, \qquad \alpha_{j,2} = \frac{1}{r_{22}^2}\left(\Delta y_j - \frac{r_{12}}{r_{11}}\Delta x_j\right)$$

$$\alpha_{j,3} = \frac{1}{r_{33}^2}\left(\Delta z_j - \frac{r_{23}}{r_{22}}\Delta y_j + \psi\Delta x_j\right), \qquad \psi = \frac{r_{12}r_{23} - r_{13}r_{22}}{r_{11}r_{22}}$$

and

$$r_{11} = \sqrt{\sum_{j=1}^{NB(0)} (\Delta x_j)^2}, \quad r_{12} = \frac{1}{r_{11}} \sum_{j=1}^{NB(0)} (\Delta x_j)(\Delta y_j)$$

$$r_{13} = \frac{1}{r_{11}} \sum_{j=1}^{NB(0)} (\Delta x_j)(\Delta z_j), \quad r_{22} = \sqrt{\sum_{j=1}^{NB(0)} (\Delta y_j)^2 - r_{12}^2}$$

$$r_{23} = \frac{1}{r_{22}} \sum_{j=1}^{NB(0)} (\Delta y_j)(\Delta z_j) - r_{12}r_{13}, \quad r_{33} = \sqrt{\sum_{j=1}^{NB(0)} (\Delta z_j)^2 - (r_{13}^2 + r_{23}^2)}$$

where NB(0) refers to neighbor nodes around "0" (0 →P).

- Implementation of the least-squares gradient reconstruction requires that six r values (r_{11}, r_{22}, r_{33}, r_{12}, r_{13}, r_{23}) be stored at each node P.
- The calculation needs to be performed only once for each node.

Gradient reconstruction using Least-Squares with weighting coefficients

- In the above least-squares method each neighbor node contributes equally to the gradient reconstruction irrespective of its distance to the central point P.
- The accuracy of the method may detorriate on highly distorted meshes.
- It is possible to give different weights to the contribution of each neighbor node to gradient reconstruction. This yield the following set of equations for 2D problems:

$$\begin{bmatrix} w_1\Delta x_1 & w_1\Delta y_1 \\ w_2\Delta x_2 & w_2\Delta y_2 \\ w_3\Delta x_3 & w_3\Delta y_3 \\ \vdots & \vdots \\ w_N\Delta x_N & w_N\Delta y_N \end{bmatrix} \begin{bmatrix} \left(\frac{\partial\phi}{\partial x}\right)_0 \\ \left(\frac{\partial\phi}{\partial y}\right)_0 \end{bmatrix} = \begin{bmatrix} w_1(\phi_1-\phi_0) \\ w_2(\phi_2-\phi_0) \\ w_3(\phi_3-\phi_0) \\ \vdots \\ w_N(\phi_N-\phi_0) \end{bmatrix}$$

where w_i is the weighting factor of neighbor node i.

- The w_i's can be selected as a function of the geometry or the solution.
- Geometrical weights can be chosen in the form

$$w_i = \frac{1}{|\mathbf{r}_i - \mathbf{r}_P|^t} \qquad t = 0,1,2.$$

- For inverse distance weighting $t = 1, \rightarrow w_i = 1/\sqrt{\Delta x_i^2 + \Delta y_i^2 + \Delta z_i^2}$
- The gradient using the weighted least-squares reconstruction becomes

$$(\nabla\phi)_0 = \begin{bmatrix} \left(\nabla^x\phi\right)_0 \\ \left(\nabla^y\phi\right)_0 \\ \left(\nabla^z\phi\right)_0 \end{bmatrix} \begin{bmatrix} \left(\frac{\partial\phi}{\partial x}\right)_0 \\ \left(\frac{\partial\phi}{\partial y}\right)_0 \\ \left(\frac{\partial\phi}{\partial z}\right)_0 \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{NB(0)} C_j^x w_j(\phi_j-\phi_0) \\ \sum_{j=1}^{NB(0)} C_j^y w_j(\phi_j-\phi_0) \\ \sum_{j=1}^{NB(0)} C_j^z w_j(\phi_j-\phi_0) \end{bmatrix}$$

where the C terms are the same as that of the unweighted method:

$$C_j^x = \alpha_{j,1} - \frac{r_{12}}{r_{11}}\alpha_{j,2} + \psi\alpha_{j,3}, \qquad C_j^y = \alpha_{j,2} - \frac{r_{23}}{r_{22}}\alpha_{j,3}, \qquad C_j^z = \alpha_{j,3}$$

- The α and r terms are

$$\alpha_{j,1} = \frac{w_j \Delta x_j}{r_{11}^2}, \qquad \alpha_{j,2} = \frac{1}{r_{22}^2}\left(w_j \Delta y_j - \frac{r_{12}}{r_{11}} w_j \Delta x_j\right)$$

$$\alpha_{j,3} = \frac{1}{r_{33}^2}\left(w_j \Delta z_j - \frac{r_{23}}{r_{22}} w_j \Delta y_j + \psi w_j \Delta x_j\right), \qquad \psi = \frac{r_{12}r_{23} - r_{13}r_{22}}{r_{11}r_{22}}$$

$$r_{11} = \sqrt{\sum_{j=1}^{NB(0)} \left(w_j \Delta x_j\right)^2}, \qquad r_{12} = \frac{1}{r_{11}} \sum_{j=1}^{NB(0)} \left(w_j \Delta x_j\right)\left(w_j \Delta y_j\right)$$

$$r_{13} = \frac{1}{r_{11}} \sum_{j=1}^{NB(0)} \left(w_j \Delta x_j\right)\left(w_j \Delta z_j\right), \qquad r_{22} = \sqrt{\sum_{j=1}^{NB(0)} \left(w_j \Delta y_j\right)^2 - r_{12}^2}$$

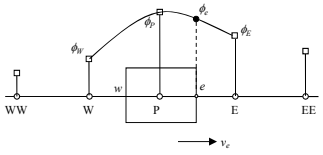
$$r_{23} = \frac{1}{r_{22}} \sum_{j=1}^{NB(0)} \left(w_j \Delta y_j\right)\left(w_j \Delta z_j\right) - r_{12}r_{13}, \qquad r_{33} = \sqrt{\sum_{j=1}^{NB(0)} \left(w_j \Delta z_j\right)^2 - \left(r_{13}^2 + r_{23}^2\right)}$$

TVD Schemes in Unstructured Grids: 1) Flux Limiters

For Cartesian grids, for $u_e > 0$, using a TVD scheme, ϕ_e may be written as

$$\phi_e = \phi_P + \frac{1}{2}\psi(r)(\phi_E - \phi_P) \qquad (11.83)$$

where $r = \dfrac{\phi_P - \phi_W}{\phi_E - \phi_P}$



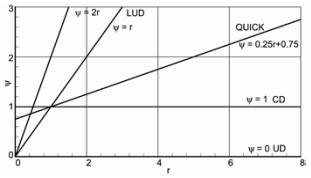
ψ = flux limiter function

$\psi = 0$ for UD scheme

$\psi = 1$ for CD scheme

$\psi = r$ for LUD scheme

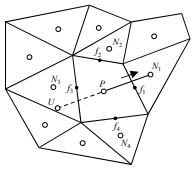
$\psi = 0.25r + 0.75$ for QUICK scheme



However, in unstructured grids, for face f_1 of cell P, the upwind point U is not available.

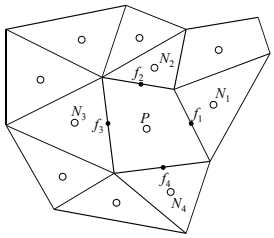
There are methods to construct U, but these are costly.

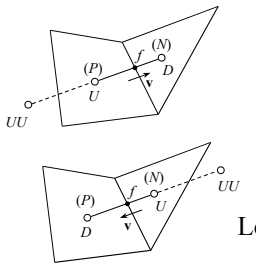
In the absence of ϕ_U, the method proposed by Darwish and Moukalled (2003) will be used



TVD Schemes in Unstructured Grids : 1) Flux Limiters

Since the index based notation used above is not suitable for unstructured grids, a more appropriate notation proposed by Darwish and Moukalled (2003) will be adopted.





U: upwind node around face *f*

D: downwind node around face *f*

UU: (virtual) upwind node far upstream

Locate *UU* such that $\mathbf{r}_{(UU)D} = 2\mathbf{r}_{UD}$

Using this notation:

$$\phi_f = \phi_U + \frac{1}{2}\psi(r_f)(\phi_D - \phi_U) \tag{11.84}$$

where $r_f = \dfrac{\phi_U - \phi_{UU}}{\phi_D - \phi_U}$ (11.85)

The main difficulty in using TVD schemes in unstructured grids lies in the need for defining a virtual node *UU*.

Exact *r* Formulation

Returning to the definition of *r*:

$$r_f = \frac{\phi_U - \phi_{UU}}{\phi_D - \phi_U} = \frac{\phi_D + (\phi_U - \phi_{UU}) - \phi_D}{\phi_D - \phi_U} = \frac{(\phi_D - \phi_{UU}) - (\phi_D - \phi_U)}{\phi_D - \phi_U} \tag{11.86}$$

Note that *D* and *U* are the nodes straddling the interface and thus are readily available.

For node *UU* we can write:

$$(\phi_D - \phi_{UU}) = \nabla\phi_U \cdot \mathbf{r}_{(UU)D} = (2\nabla\phi_U \cdot \mathbf{r}_{UD}) \tag{11.87}$$

since $\mathbf{r}_{(UU)D} = 2\mathbf{r}_{UD}$ (*U* is at the center of the (*UU*)*D* segment). Other positions of *UU* could also be chosen but with a loss of accuracy as the nodal gradient yields a second order accuracy only when the difference is centered at *U*.

The formulation of r_f becomes

$$r_f = \frac{(2\nabla\phi_U \cdot \mathbf{r}_{UD}) - (\phi_D - \phi_U)}{\phi_D - \phi_U} = \frac{(2\nabla\phi_U \cdot \mathbf{r}_{UD})}{\phi_D - \phi_U} - 1 \tag{11.88}$$

After calculating r_f from Eq. (11.88) find $\psi(r_f)$ from a TVD scheme (chapter 5) and then calculate ϕ_f from Eq. (11.84).

Exact *r* Formulation

Note that this *r*-formulation can also be used for any high order scheme without TVD. For example for QUICK scheme:

$$\psi(r_f) = 0.25 r_f + 0.75$$

Then, the application of high order convection schemes with TVD in a deferred correction manner, is added as a source term to S_{dc} in accordance with Eq. (11.46) as

$$S^{dc} = -\left[\sum_{f=1}^{n_f(P)} \left(\dot{m}_f \phi_f^H - \max(\dot{m}_f, 0.)\phi_P - \min(\dot{m}_f, 0.)\phi_N\right)\right]^{n-1} \qquad (11.89)$$

where, ϕ_f^H is the ϕ_f value computed from Eq. (11.84).

TVD Schemes in Unstructured Grids: 2) Gradient Limiters

- For structured grids flux limiters are used. For unstructured grids, the **gradient or slope limiters are used**. Instead of limiting the flux at the cell face, in this method the gradient $\nabla\phi_P$ at cell center P, is limited.
- Barth and Jespersen (1989) introduced the first gradient limiter for unstructured grids.
- The scheme consists of finding the value of ϕ in a CV by limiting the gradient such that no extrema of ϕ is formed:

$$\phi(\mathbf{r}) = \phi_P + \Phi \nabla\phi_P \cdot (\mathbf{r} - \mathbf{r}_P) \qquad (11.90)$$

- Φ = gradient limiter for the control volume at P, $\mathbf{r}$ = position vector.
- That is, to prevent formation of extrema in the CV, Φ limits the gradient such that the value of $\phi(\mathbf{r})$ calculated in the CV (and at the faces) should satisfy the following:
- $\phi(\mathbf{r}) < \max(\phi_P, \phi_{Ni})$, $i = 1, n$ and
- $\phi(\mathbf{r}) > \min(\phi_P, \phi_{Ni})$, $i = 1, n$ where n is the number of neighbors of P

- Using the limited gradient $\Phi\nabla\phi$, then the value of ϕ at any cell face is found from

$$\phi_f = \phi_P + \Phi\nabla\phi_P \cdot \mathbf{r}_{Pf} \qquad (11.91)$$

where $\mathbf{r}_{Pf}$ is the position vector from P to the face center f.

- Point f is called the reconstruction point. (ϕ_f is reconstructed from its gradient at P)
- However it should be noted that reconstruction points may also be chosen to be the vertices of the face, which is common in vertex-based unstructured grid methods.
- The advantage of gradient limiters over the flux limiters is that, not only the convection fluxes, but diffusion fluxes are also limited, since these fluxes contain gradients of ϕ.

Barth and Jespersen's gradient limiter

- The following procedure is used by **Barth and Jespersen**:

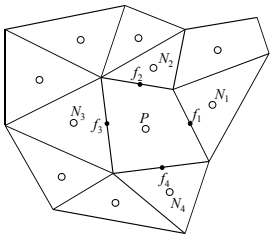
1) Find the largest negative ($\Delta_{\min} = \phi_{\min} - \phi_P$) and positive ($\Delta_{\max} = \phi_{\max} - \phi_P$) difference between the neighbors and the current CV.

2) Compute the value of ϕ_{fi} for face i from the unlimited expression

$$\phi_{fi} = \phi_P + \nabla\phi_P \cdot \mathbf{r}_{Pfi} \qquad (11.92)$$

3) Next, define the ratio y_i for each face i of CV as:

$$y_i = \begin{cases} \dfrac{\Delta_{\max}}{\Delta_{fi}} & \text{if } \Delta_{fi} > 0 \\ \dfrac{\Delta_{\min}}{\Delta_{fi}} & \text{if } \Delta_{fi} < 0 \\ 1 & \text{if } \Delta_{fi} = 0 \end{cases} \qquad (11.93)$$



where

$$\Delta_{\max} = \phi_{\max} - \phi_P, \quad \Delta_{\min} = \phi_{\min} - \phi_P, \quad \Delta_{fi} = \phi_{fi} - \phi_P$$

$$\phi_{\max} = \max(\phi_P, \phi_{N_i}), \quad i = 1, n$$

$$\phi_{\min} = \min(\phi_P, \phi_{N_i}), \quad i = 1, n$$

3) Compute a maximum allowable value of Φ_{fi} for each face i of CV:

$$\Phi_{fi} = \min(1, y_i)$$

4) Select

$$\Phi_{BJ} = \min(\Phi_{fi}), \quad i = 1, n \tag{11.94}$$

for the node P, where n = number of faces of the CV

- That is, the gradient limiter of Barth-Jespersen, Φ_{BJ}, is defined as the minimum of all the neighboring face limiters Φ_{fi}.
- A compact formulation of Barth-Jespersen's limiter is given by Wang (1998) as

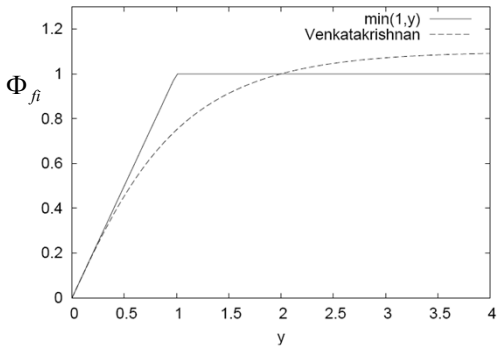
$$\Phi_{BJ} = \min\left[1, \left(\frac{\max\left|\phi_{Ni} - \phi_P\right|_{i=1,n}}{\max\left|\phi_{fi} - \phi_P\right|_{i=1,n}}\right), \left(\frac{\min\left|\phi_{Ni} - \phi_P\right|_{i=1,n}}{\min\left|\phi_{fi} - \phi_P\right|_{i=1,n}}\right)\right] \tag{11.95}$$

- The function Φ_{fi} proposed by Barth and Jespersen is non-differentiable. This adversely affects the convergence of the solver.
- Venkatakrishnan, (1995) proposed modifications that replace the discontinuous limiter function with a smoothly varying function.
- While the new limiter does not strictly enforce monotonicity of the reconstructed fluxes, it has better convergence properties.

$$\Phi_{fi}(y) = \frac{y^2 + 2y}{y^2 + y + 2} \tag{11.96}$$

where $y = \dfrac{\Delta_{\max}}{\Delta_{fi}}$ if $\Delta_{fi} > 0$ and $y = \dfrac{\Delta_{\min}}{\Delta_{fi}}$ if $\Delta_{fi} < 0$

- Although this new function is differentiable, it has the disadvantage of reducing the gradient even in regions where no new extrema are formed (i.e. for $y < 2$ as can be seen in the following figure).



Venkatakrishnan's smooth approximation to min(1, y)

However, for smooth flows (where ϕ is nearly uniform) on a uniform mesh, y is expected to be $\cong 2$ and this method introduces an error which is on the order of truncation.
To avoid applying the limiter in regions of nearly uniform flow Venkatakrishnan introduced a modification to the method.

- The modification eliminates the effect of the limiter when

$$\phi_f - \phi_P \le (K\Delta x)^{1.5} \qquad (11.97)$$

where Δx is a characteristic length for the CV ($\Delta x = (V_{CV})^{1/3}$)

- K is a tunable parameter (K = 0.1–2, but usually K = 0.3 is used).
- The modified limiter of Venkatakrishnan is

$$\Phi_{fi} = \begin{cases} \dfrac{1}{\Delta_{fi}} \dfrac{(\Delta_{\max}^2 + \varepsilon^2)\Delta_{fi} + 2\Delta_{fi}^2 \Delta_{\max}}{\Delta_{\max}^2 + 2\Delta_{fi}^2 + \Delta_{\max}\Delta_{fi} + \varepsilon^2} & \text{if } \Delta_{fi} > 0 \\ \dfrac{1}{\Delta_{fi}} \dfrac{(\Delta_{\min}^2 + \varepsilon^2)\Delta_{fi} + 2\Delta_{fi}^2 \Delta_{\min}}{\Delta_{\min}^2 + 2\Delta_{fi}^2 + \Delta_{\min}\Delta_{fi} + \varepsilon^2} & \text{if } \Delta_{fi} < 0 \\ 1 & \text{if } \Delta_{fi} = 0 \end{cases} \qquad (11.98)$$

where $\varepsilon^2 = (K\Delta x)^3$

- With $K > 0$, this modification does not maintain strict monotonicity.

Limiter of Michalac and Ollivier-Gooch

- Venkatakrishnan's limiter does not provide sufficient accuracy even in smooth regions without any local extrema.
- While the Barth-Jespersen limiter does satisfy these conditions, convergence to steady state solution is difficult due to lack of differentiability of min(1, y).
- Michalac and Olliver-Gooch (2008) proposed a new approximation for min(1, y) in the limiter function of Barth-Jespersen that will be monotone and differentiable:

$$\Phi_{fi} = \begin{cases} P(y) & y < y_t \\ 1 & y \geq y_t \end{cases} \qquad (11.99)$$

where $1 < y_t < 2$ is a threshold and $P(y)$ is a polynomial satisfying

$$P\big|_0 = 0, \quad P\big|_{y_t} = 1$$

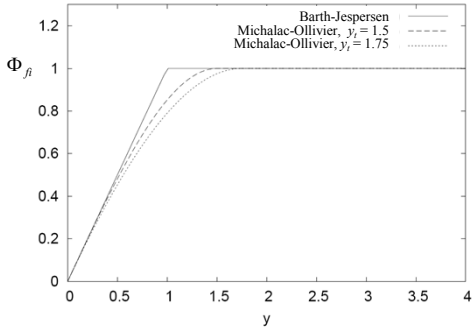
$$\left.\frac{dP}{dy}\right|_0 = 1, \quad \left.\frac{dP}{dy}\right|_{y_t} = 0$$

A third degree polynomial of the from

$$P(y) = a_3 y^3 + a_2 y^2 + a_1 y + a_0 \qquad (11.100)$$

can satisfy the above requirements with coefficients:

$$a_3 = (y_t - 2)/y_t^3, \quad a_2 = -(1 + 3a_3 y_t^2)/(2y_t), \quad a_1 = 1, \quad a_0 = 0$$



The resulting limiters for $y_t = 1.5$, 1.75 are plotted in the above figure. For $y_t = 1.5$ accuracy is better. For $y_t = 1.75$ convergence is better.

Modifications to Limiter in uniform regions

- In regions of flow which are nearly uniform, extrema will occur frequently in the transient solution.
- Therefore, it is desirable to switch-off the limiter in those regions.
- Michalac and Ollivier-Gooch suggest to switch the limiter off when

$$\Delta\phi \equiv (\Delta_{\max} - \Delta_{\min}) < (K\Delta x)^{3/2} \qquad (11.101)$$

- To maintain differentiability, a modified limiter, $\Phi_{BJ\text{-}mod}$, is proposed:

$$\Phi_{BJ-mod} = \sigma + (1-\sigma)\Phi_{BJ} \qquad (11.102)$$

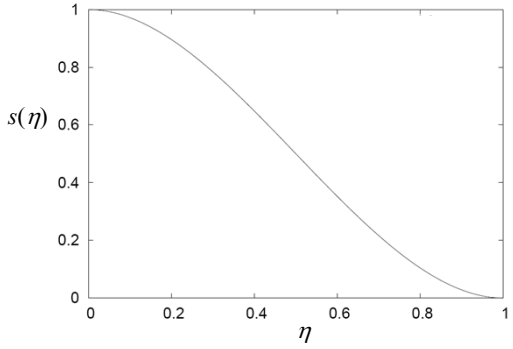
where Φ_{BJ} is Barth-Jespersen's limiter calculated from Eqn. (11.94) and σ is

$$\sigma = \begin{cases} 1 & \text{for } (\Delta\phi)^2 \le (K\Delta x)^3 \\ s(\eta) & \text{for } (K\Delta x)^3 < (\Delta\phi)^2 < 2(K\Delta x)^3 \\ 0 & \text{for } (\Delta\phi)^2 \ge 2(K\Delta x)^3 \end{cases} \qquad (11.103)$$

where the transition function $s(\eta)$ is

$$s(\eta) = 2\eta^3 - 3\eta^2 + 1, \qquad \eta = \frac{(\Delta\phi)^2 - (K\Delta x)^3}{(K\Delta x)^3} \qquad (11.104)$$

The transition function $s(\eta)$



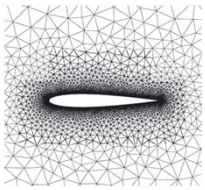
The transition function $s(\eta)$ is used to smoothly disable the limiter in nearly uniform meshes.
In this way, the cells with $\sigma = 1$ have $\Phi_{new} = 1$.
As a result Φ need not be calculated in nearly uniform flow regions since from (11.102)

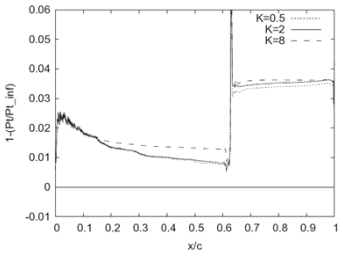
$$\Phi_{BJ-mod} = \sigma + (1-\sigma)\Phi_{BJ}$$

This decreases the computer time.

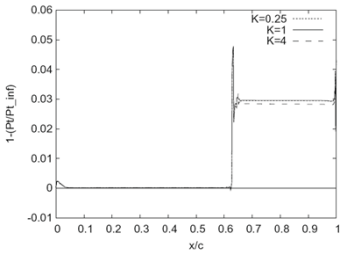
Sensitivity of results to K

Consider transonic flow at Mach 0.8 across an airfoil shown in the figure. (Michalac and Olliver-Gooch (2009))
The sensitivity of total pressure loss across the shock near the leading edge of the airfoil, to the tuning parameter K, is shown in the figure below.





Venkatakrishnan's limiter, Φ_V



Barth-Jespersen's modified limiter, $\Phi_{BJ\text{-}mod}$

The modified limiter exhibits almost no sensitivity to K before the shock.
Downstream the shock the results using $K = 0.25$ and $K = 1$ are indistinguishable.

Unstructured Mesh generation

- Commercial CFD packages such as
- FLUENT
- CFX
- STAR-CD
- PHOENICS
- FLOW3D

have their own mesh generation tools together with their geometry modelers.

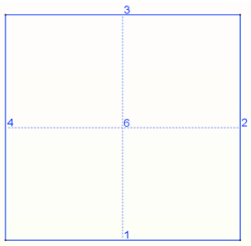
- They also have facilities to import data from solid modeler packages such as
- PARASOLID, PRO/ENGINEER
- SOLID EDGE, SOLID WORKS
- UNIGRAPHICS

- There are also free mesh generation packages such as
- GMSH
- SALOME
- Some of these free mesh generation packages can also import geometry files from solid modelers.
- A geometric model of the solution domain is first developed.
- Then, meshes are created on this geometric model.
- Salome seems to have a better GUI (graphics user interface) and is more user friendly than GMSH for the time being (as of 2015).
- The mesh file created by the mesh generator software consists of two parts which give information for nodes and elements. That is:

  1) x, y and z coordinates of each node in the mesh

  2) list of nodes of each element together with its element type and region (zone) number.

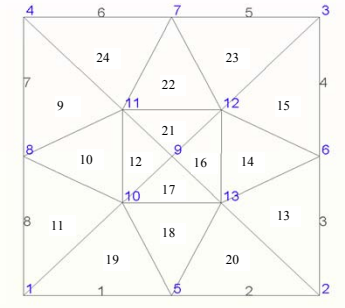
An example drawing created by GMSH



Drawing 1

The drawing of a 2-D square domain created by GMSH is shown in the figure.
The physical region number of each boundary is denoted by numbers from 1 to 4.
1 → bottom, 2 → right, 3 → top, 4 → left boundaries.
The surface is indicated by dashed lines and has a physical region number of 6.

The mesh of drawing 1 created by GMSH




```
$Nodes
13
1 0 0 0
2 1 0 0
3 1 1 0
4 0 1 0
5 0.4999999999986718 0 0
6 1 0.4999999999986718 0
7 0.5000000000013305 1 0
8 0 0.5000000000013305 0
9 0.5013711139750536 0.4999926352623462 0
10 0.3339866124247404 0.3331457930147393 0
11 0.3343386761842688 0.6672116118139605 0
12 0.6682287346435727 0.6666784466453495 0
13 0.6678219567120327 0.3329201164922819 0
$EndNodes
```

```
$Elements
24
1 1 2 1 1 1 5
2 1 2 1 1 5 2
3 1 2 2 2 2 6
4 1 2 2 2 6 3
5 1 2 3 3 3 7
6 1 2 3 3 7 4
7 1 2 4 4 4 8
8 1 2 4 4 8 1
9 2 2 99 6 4 8 11
10 2 2 99 6 8 10 11
11 2 2 99 6 8 1 10
12 2 2 99 6 11 10 9
13 2 2 99 6 2 6 13
14 2 2 99 6 6 12 13
15 2 2 99 6 6 3 12
16 2 2 99 6 13 12 9
17 2 2 99 6 9 10 13
18 2 2 99 6 10 5 13
19 2 2 99 6 10 1 5
20 2 2 99 6 13 5 2
21 2 2 99 6 9 12 11
22 2 2 99 6 12 7 11
23 2 2 99 6 12 3 7
24 2 2 99 6 11 7 4
$EndElements
```

There are 13 nodes (blue colors) and 24 elements.
First 8 elements are boundary elements (line elements, elm type = 1).
Elements 9-24 are interior elements (triangular elmnts, elm type = 2).
In nodes data: 1st column is node number, columns 2-4 are x, y, z coordinates of nodes.
In elements data: first column is element number, column 2 → elm type, column 3 → number of tags, columns 4-5 → tags, (tag1 → physical region number, tag 2 → elementary region number), the rest of columns (columns 6-7 or 6-8) → nodes of element.

Unstructured Grid Element Numbering Conventions

- For an unstructured grid, the element connectivity cannot be easily built, so this additional information is generally added to the data file.
- The element information typically includes the element type or shape, and the list of nodes for each element.
- Unstructured grid element numbering convention of CGNS (CFD General Notation System) will be given here.
- CGNS provides a general, portable, and extensible standard for the storage and retrieval of CFD analysis data.
- In an unstructured zone, the nodes are ordered from 1 to N, where N is the number of nodes in the zone.
- An element is defined as a group of one or more nodes, where each node is represented by its index.
- The elements are indexed from 1 to M within a zone, where M is the total number of elements defined for the zone.

- CGNS supports eight element shapes – points, lines, triangles, quadrangles, tetrahedra, pentahedra, pyramids, and hexahedra.
- Elements describing a volume are referred to as 3-D elements.
- Those defining a surface are 2-D elements.
- Line and point elements are called 1-D and 0-D elements, respectively.
- In a 3-D unstructured mesh, the cells are defined using 3-D elements, while the boundary patches may be described using 2-D elements.
- Each element shape may have a different number of nodes, depending on whether linear or quadratic interpolation is used.
- Therefore the name of each type of element is composed of two parts; the first part identifies the element shape, and the second part the number of nodes.

Element types in CGNS

Element Types in CGNS

Dimensionality	Shape	Linear Interpolation	Quadratic Interpolation
0-D	Point	NODE	NODE
1-D	Line	BAR_2	BAR_3
2-D	Triangle	TRI_3	TRI_6
	Quadrangle	QUAD_4	QUAD_8, QUAD_9
3-D	Tetrahedron	TETRA_4	TETRA_10
	Pyramid	PYRA_5	PYRA_14
	Pentahedron	PENTA_6	PENTA_15, PENTA_18
	Hexahedron	HEXA_8	HEXA_20, HEXA_27

1-D line elements

BAR_2

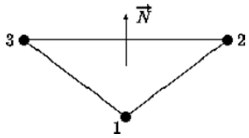
BAR_3

Face Definition

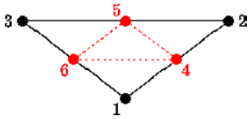
Oriented edge	**Corner nodes**	**Mid node**
E1	N1,N2	N3

2-D (surface) elements

CGNS supports two shapes of 2-D elements - triangles and quadrangles.



TRI_3



TRI_6

Edge Definition

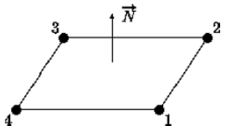
Oriented edges	**Corner nodes**	**Mid node**
E1	N1,N2	N4
E2	N2,N3	N5
E3	N3,N1	N6

Face Definition

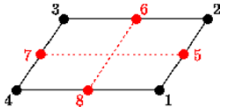
Face	**Corner nodes**	**Mid-edge nodes**	**Oriented edges**
F1	N1,N2,N3	N4,N5,N6	E1,E2,E3

N1,...,N6	Grid point identification number. Integer ≥ 0 or blank, and no two values may be the same. Grid points N1, N2, and N3 are in consecutive order about the triangle.
E1,E2,E3	Edge identification number.
F1	Face identification number.

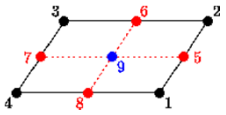
Quadrilateral elements



QUAD_4



QUAD_8



QUAD_9

Edge Definition

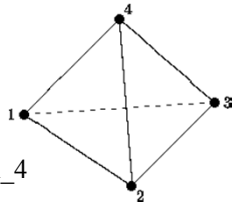
Oriented edges		Corner nodes		Mid node
E1		N1,N2		N5
E2		N2,N3		N6
E3		N3,N4		N7
E4		N4,N1		N8

Face Definition

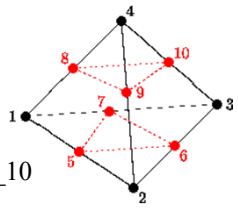
Face		Corner nodes		Mid-edge nodes		Mid-face node		Oriented edges
F1		N1,N2,N3,N4		N5,N6,N7,N8		N9		E1,E2,E3,E4

3-D volume elements

CGNS supports four different shapes of 3-D elements - tetrahedra, pyramids, pentahedra, and hexahedra.



TETRA_4



TETRA_10

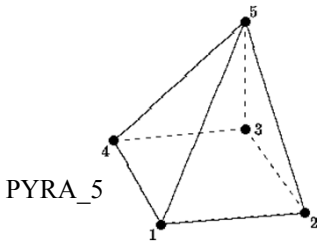
Edge Definition

Oriented edges	Corner nodes	Mid node
E1	N1,N2	N5
E2	N2,N3	N6
E3	N3,N1	N7
E4	N1,N4	N8
E5	N2,N4	N9
E6	N3,N4	N10

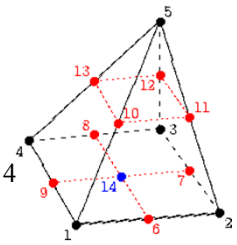
Face Definition

Face	Corner nodes		Mid-edge nodes		Oriented edges
F1	N1,N3,N2		N7,N6, N5		-E3,-E2,-E1
F2	N1,N2,N4		N5,N9, N8		E1, E5,-E4
F3	N2,N3,N4		N6,N10,N9		E2, E6,-E5
F4	N3,N1,N4		N7,N8, N10		E3, E4,-E6

Pyramid elements



PYRA_5



PYRA_14

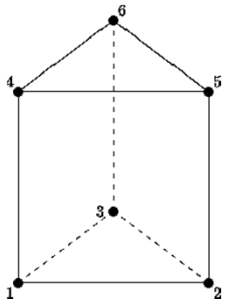
Edge Definition

Oriented edges	Corner nodes	Mid node
E1	N1,N2	N6
E2	N2,N3	N7
E3	N3,N4	N8
E4	N4,N1	N9
E5	N1,N5	N10
E6	N2,N5	N11
E7	N3,N5	N12
E8	N4,N5	N13

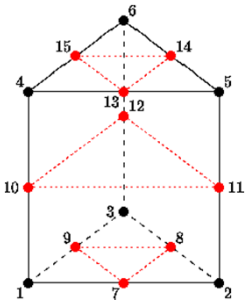
Face Definition

Face	Corner nodes	Mid-edge nodes	Mid-face node	Oriented edges
F1	N1,N4,N3,N2	N9,N8, N7, N6	N14	-E4,-E3,-E2,-E1
F2	N1,N2,N5	N6,N11,N10		E1, E6,-E5
F3	N2,N3,N5	N7,N12,N11		E2, E7,-E6
F4	N3,N4,N5	N8,N13,N12		E3, E8,-E7
F5	N4,N1,N5	N9,N10,N13		E4, E5,-E8

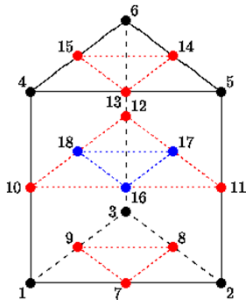
Pentahedral elements



PENTA_6



PENTA_15



PENTA_18

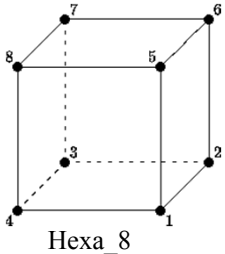
Edge Definition

Oriented edges	Corner nodes	Mid node
E1	N1,N2	N7
E2	N2,N3	N8
E3	N3,N1	N9
E4	N1,N4	N10
E5	N2,N5	N11
E6	N3,N6	N12
E7	N4,N5	N13
E8	N5,N6	N14
E9	N6,N4	N15

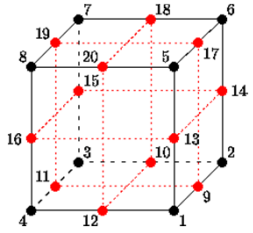
Face Definition

Face	Corner nodes	Mid-edge nodes	Mid-face node	Oriented edges
F1	N1,N2,N5,N4	N7, N11,N13,N10	N16	E1, E5,-E7,-E4
F2	N2,N3,N6,N5	N8, N12,N14,N11	N17	E2, E6,-E8,-E5
F3	N3,N1,N4,N6	N9, N10,N15,N12	N18	E3, E4,-E9,-E6
F4	N1,N3,N2	N9, N8, N7		-E3,-E2,-E1
F5	N4,N5,N6	N13,N14,N15		E7, E8, E9

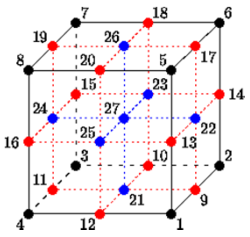
Hexahedral elements



Hexa_8



Hexa_20



Hexa_27

Edge Definition

Oriented edges	Corner nodes	Mid node
E1	N1,N2	N9
E2	N2,N3	N10
E3	N3,N4	N11
E4	N4,N1	N12
E5	N1,N5	N13
E6	N2,N6	N14
E7	N3,N7	N15
E8	N4,N8	N16
E9	N5,N6	N17
E10	N6,N7	N18
E11	N7,N8	N19
E12	N8,N5	N20

Face Definition

Face	Corner nodes	Mid-edge nodes	Mid-face node	Oriented edges
F1	N1,N4,N3,N2	N12,N11,N10,N9	N21	-E4,-E3, -E2, -E1
F2	N1,N2,N6,N5	N9, N14,N17,N13	N22	E1, E6, -E9, -E5
F3	N2,N3,N7,N6	N10,N15,N18,N14	N23	E2, E7, -E10,-E6
F4	N3,N4,N8,N7	N11,N16,N19,N15	N24	E3, E8, -E11,-E7
F5	N1,N5,N8,N4	N13,N20,N16,N12	N25	E5,-E12,-E8, E4
F6	N5,N6,N7,N8	N17,N18,N19,N20	N26	E9, E10, E11, E12

References in addition to that of the textbook

- Venkatakrishnan, V. (1993), On the accuracy of limiters and convergence to steady-state solutions, *AIAA paper* 93-0880, Jan.
- Barth, T.J. and Jespersen, D.C. (1989). The design and application of upwind schemes on unstructured meshes, *AIAA paper* 89-0366, Jan.
- Michalak M. and Ollivier-Gooch C. (2008), Limiters for unstructured higher-order accurate solutions of the Euler equations, In *Forty-sixth Aerospace Sciences Meeting*.
- Michalak M. and Ollivier-Gooch C. (2009), Accuracy preserving limiter for the high-order accurate solution of the Euler equations, *Journal of Computational Physics*, vol. 228, pp. 8693-8711.
- Wang, Z.J. (1998), A quadtree-based adaptive Cartesian/quad grid flow solver for Navier-Stokes equations, *Computers and Fluids*, Vol. 27, pp. 529-549.